

国际信息工程先进技术译丛

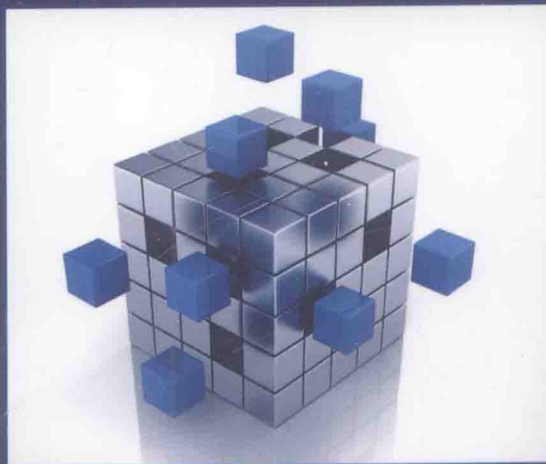
WILEY

工业关键系统的形式化方法：应用综述

Formal Methods for Industrial Critical Systems: A Survey of Applications

[意] Stefania Gnesi
Tiziana Margaria 著

靳添絮 连晓峰 等译







机械工业出版社
CHINA MACHINE PRESS

国际信息工程先进技术译丛

工业关键系统的形式化方法：应用综述

［意］ Stefania Gnesi
Tiziana Margaria 著

靳添絮　连晓峰　等译





机械工业出版社

图书在版编目（CIP）数据

工业关键系统的形式化方法：应用综述/（意）格涅斯（Gnesi, S.），（意）玛格丽特（Margaria, T.）著；靳添絮等译．—北京：机械工业出版社，2014.12

（国际信息工程先进技术译丛）

书名原文：Formal methods for industrial critical systems: a survey of applications

ISBN 978-7-111-48521-6

Ⅰ.①工… Ⅱ.①格…②玛…③靳… Ⅲ.①计算机技术－应用－工业－自动控制系统－研究 Ⅳ.①TP273

中国版本图书馆 CIP 数据核字（2014）第 266002 号

机械工业出版社（北京市百万庄大街 22 号 邮政编码 100037）

策划编辑：顾 谦 责任编辑：顾 谦

版式设计：霍永明 责任校对：樊钟英

封面设计：马精明 责任印制：李 洋

北京市四季青双青印刷厂印刷

2015 年 1 月第 1 版第 1 次印刷

169mm×239mm · 15.25 印张 · 280 千字

0 001－2 600 册

标准书号：ISBN 978-7-111-48521-6

定价：69.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务	网络服务
服务咨询热线：(010)88361066	机 工 官 网：www.cmpbook.com
读者购书热线：(010)68326294	机 工 官 博：weibo.com/cmp1952
(010)88379203	教育服务网：www.cmpedu.com
封面无防伪标均为盗版	金 书 网：www.golden-book.com

形式化方法以数学为基础，其目标是建立精确的、无二义性的语义，对系统开发的各个阶段进行有效的描述，使系统的结构具有先天的合理性、正确性和良好的维护性，能较好地满足用户需求。本书记录和展示了作者关于形式化方法如何在工业关键系统中进行应用的研究成果。

本书分为6部分：第1部分是概述；第2部分致力于介绍建模范例；第3部分介绍了包括形式化方法和相关工具的使用以及应用程序在实际系统领域的发展；第4部分则向读者展示了形式化方法在通信系统中的发展和成果；第5部分则介绍了形式化方法在互联网和在线服务方面的应用；而在第6部分则介绍了实时应用程序的形式化方法。

本书可用作高等院校计算机科学、自动化相关专业本科生、研究生以及教师的参考用书，也可作为业内专业人士的参考书。

译 者 序

形式化方法以数学为基础，对系统开发的各个阶段进行有效的描述，是有效验证系统设计和开发正确性的重要手段之一。让已经被普遍应用于测试方法复杂且对安全性有很高要求的控制系统的形式化方法更好地融入工业中，并使得它们在那里可以发挥最大的作用，这也是译者翻译本书的初衷。

本书作者是 Stefania Gnesi 和 Tiziana Margaria。其中，Stefania Gnesi 之前在佛罗伦萨大学任教，主讲针对软件系统分析和规范的方法和工具，现在是意大利比萨 ISTI-CNR 的一位形式化方法和工具实验室的领导。而 Tiziana Margaria 则是波茨坦大学数学和自然科学学院的一位教授，在那里她负责信息学院的服务和软件工程学科，也曾在德国哥廷根（Göttingen）大学、多特蒙德工业大学、帕绍大学，以及瑞典和意大利的大学游历过。应该说，作者在形式化方法在工业关键系统应用方面是有很深研究的。

本书分为 6 部分：第 1 部分是概述；第 2 部分致力于介绍建模范例；第 3 部分介绍了包括形式化方法和相关工具的使用以及应用程序在实际系统领域的发展；第 4 部分则向读者展示了形式化方法在通信系统的发展和成果；第 5 部分则介绍了形式化方法在互联网和在线服务方面的应用；而在第 6 部分则介绍了实时应用程序的形式化方法。

本书第 1 ~ 3 章由靳添絮翻译，第 4 ~ 7 章由连晓峰翻译，第 8 章由董美华、胡冰川、班岚和金成学翻译，第 9 章由胡波、周锐、王佩荣和潘媛翻译，第 10 章由苑昆、郑舒阳、贾琦和毋冬翻译，第 11 章由陆亚灵、郭晓钰和王炜伊翻译。全书由靳添絮审校整理，并对原书中的错误进行修正。

限于译者的经验和水平，书中难免存在缺点和错误，敬请广大读者批评指正。

译 者

原　书　序

——**Mike Hinchey**

正如我们所知，作为许多信用先驱动机的第一台计算机是整个19世纪航运业的一个重大问题。在此期间出现的对数表对该行业至关重要，通常包含造成船只、货物和生命损失的简单但显著的错误。一般认为Babbage差动机可体现计算机系统许多概念标准（包括存储、程序甚至类似于现代激光打印机工作原理的打印机设计）。目的是实现自动打印航运业常用的表格并降低不准确性。

取决于计算位置的数据*正确性*对航运业尤为关键。“正确性”的概念自从计算机科学真正成立以来就一直受到困惑。在第一台实用计算机（正如现在所知）出现之前，图灵（Turing）就一直关注着20世纪30年代出现的问题。计算机先驱，如Zuse和Wilkes早就意识到，正确性将是需要解决的重要问题。

20世纪60年以来或自从具有现代电子计算机以来，可靠性和可信性等相关问题就一直受到安全、保护、性能以及许多其他问题的高度关注。提出形式化方法（Formal methods）（先于现代计算机本身的一个术语）来解决软件系统和硬件系统中的正确性问题以及涉及的其他相关问题。

对于确认“形式方法学”的学者，这是一个极大的进步，并欣慰地看到在该领域取得显著进展以及形式化方法对关键应用领域的极大贡献。然而，在现实中，形式化方法仍没有在实践中得到所期望的广泛应用，许多人认为该方法并未形成规模，成本高，且难以理解和应用。

形式化方法研究人员认为主要关注于教学、开发更多有用符号以及更好(集成）的支持工具，强调系统的某一方面而不是整个系统（即所谓的形式化方法作用），建立用户社区，并鼓励在现实生活中应用形式化方法。在关键的工业领域，形式化方法已得到广泛应用。值得注意的是，“关键”的定义已发生变化，意味着不仅仅是生命或财产损失，或违反安全以及故障所产生的后果，而且在商业意义上，还意味着金融损失、丧失竞争力或声誉受损。

从事工业关键系统中的形式化方法（FMICS）研究的是从1992年运行至今的运行时间最长的欧洲信息与数学研究联盟工作组（ERCIM)。该工作组由超过12个ERCIM合作伙伴以及欧洲其他几十个相关研究组组成，致力于提高基于形式化方法的技术，并鼓励通过技术转让激励形式化方法在关键工业中的应用。

本书汇集了该工作组优秀研究成果以及在关键工业中的应用实例，如航空、

航天、铁路信号（已成为形式化方法技术的一个主要驱动行业）等领域。尽管本书从规范到实现和校验等各个方面讨论了形式化方法，但重点在于模型校验，反映过去10年左右时间内在工业应用中的显著进步与成功应用。

应用结果表明形式化方法在工业关键系统中的正确性。各位作者都是各自领域的专家，但不应该低估在将形式化方法引入行业中的极大困难，这种信息非常简单：对于特定应用领域和特定关键工业，形式化方法将会继续存在。

Mike Hinchey
勒罗（Lero），利默里克（Limerick）

——Alessandro Fantechi 和 Pedro Merino

本书具有很长历史，这是ERCIM中FMICS研究组的部分历史，这是该联盟中时间最久的活跃工作组。

FMICS工作组致力于形式化校验技术的发展，并开发如国际联合项目、校验相关和形式化方面的软件，并从1996年开始，举办FMICS年度研讨会。

这些活动大大促进了关于目前进行的确定最有效的形式化开发和校验技术的科学研讨，并在工业应用方面有着敏锐观察。FMICS社会成员大多与行业联系紧密，由此直接对过去10年内在工业关键系统中发展缓慢但不断引入的形式化方法作出巨大贡献。

本书的出版来源于2004年在Aix Les Bain举行的研讨会。形式化方法的不断发展，特别是由于工具性能不断完善的越来越多模型校验技术，以及近年来出现的基于模型的设计等日益增长的重要性，使之在一本书中难以全部展现当前工业应用中的发展现状。因此，本书的内容在过去几年内不断变化。

作为FMICS工作组的总负责人，为此特别感谢本书编辑，他们成功收集了该研究领域不断发展，以及在工业生产中日益增长的软件和计算机控制系统应用的文献资料。因此，我们相信这是形式化方法在大量不同领域得到应用的见证。

Alessandro Fantechi 和 Pedro Merino
FMICS 工作组主席

原书前言

目前，形式化方法的必要性已作为设计过程中不可缺少的环节，在工业安全关键系统中得到广泛认可。在更通用的定义中，“形式化方法”一词包括了所有具有精确数学语义以及以形式化方式描述系统行为的相关分析方法的符号。

在过去十几年内，出现了许多形式化方法：声明法、并发和移动系统的过程演算法以及相关语言等其他方法。尽管这些形式化方法的优点不可否认，但实践经验表明，每个方法都特别适用于处理系统某些方面。因此，设计一个理想的复杂工业系统需要多个形式化方法的专家从不同角度来描述和分析系统。

本书的目的主要是提供一种目前工业关键系统设计中主流形式化方法的全面介绍。本书有 3 个目标：减轻形式化方法的学习负担，这也是在工业应用中的一个主要缺点；帮助设计人员选择采用最适合其系统的形式化方法；介绍关键系统分析的先进技术和工具。

本书分为 6 部分。第 1 部分为概述和发展现状。第 2 部分专门介绍建模范例。其中第 2 章是关于同步数据流语言 LUSTRE 及其在 SCADE 工具集中的产业转移；第 3 章介绍了群智能的基本概念，这在各种不同应用领域如医疗、生物信息学、军事/防御、监控，甚至互联网电视广播中得到广泛应用。讨论了适用于基于群智能的系统中形式化方法的具体要求。

第 3 部分包括有关在实际系统中形式化方法的应用及其相关工具的开发应用。其中第 4 章主要是关于交通运输系统，介绍了在当前工业应用中铁路信号形式化方法的综述；第 5 章介绍了模型校验技术在航空电子领域中的应用。

第 4 部分介绍了形式化方法在通信系统中的应用。其中在第 6 章中阐述了如何应用形式化方法来提高主动服务的可靠性，尤其是重点关注代码移植、路由信息、高度重配置、服务间交互或安全策略等方面，包括互联网和在线服务；第 7 章介绍了概率模型校验的应用，特别是利用概率定时自动机进行通信协议，并重点进行了工业相关的 IEEE 802.3（CSMA/CD）案例分析。

第 5 部分涵盖了互联网与在线服务。其中第 8 章介绍了如何利用模型首先描述和验证在线分布式决策系统中的单变量，设计为一个大型协作模型的集合及自动机学习用于确定实现后实际系统的集体应急行为；第 9 章描述了利用模型校验来验证具有发布/订阅通知服务的群件系统中的用户意识。

第 6 部分介绍了运行时形式化方法的应用。其中第 10 章中，介绍了测试和测试控制表示版本 3（TTCN-3），并应用于 Web 服务测试；第 11 章对实际中自

动机学习以及其面临的主要挑战、改进以及可能的解决方案进行综述，并进行案例分析，以表明理论研究主动学习技术可得到强大应用，从而成为实际系统开发中的重要工具。

尽管意识到本书不能详尽、全面地介绍形式化方法在工业中的应用，但相信并希望读者能从中体会到该方法可能在实践应用中不断提高和促进。

Stefania Gnesi

Tiziana Margaria

目　　录

第3部分　交通运输系统

第4部分　通信系统

第5部分　互联网与在线服务

第6部分 运行时：测试和模型学习

第1部分

概述和发展现状

第1章　形式化方法：应用{逻辑关系，理论}的计算机科学

Diego Latella
ISTI-CNR，比萨，意大利

1.1　概述

对形式化方法（FM）进行一个全面定义是一项困难而又有风险的任务，这是因为这是一项仍在进行的研究课题，其中形式化方法的多个方面，包括从基础理论到单纯应用，都还在不断地进化和演变。然而，为了解决在本书中处理问题而引发讨论的各种术语，同时也出于本书的目的，将对所有语法、语义及其相关的理论和分析方法的符号进行精确地数学定义，并结合自动（软件）工具以一种形式化方法来描述和推理（计算机）系统行为。FM在计算机工程以及范围更广的系统工程中具有重要作用，同时还需考虑人机交互作用：

> 所有工程学科都是采用基于数学的符号和方法得到发展的。研究模型的‘形式化方法’并试图确定和开发一种有助于创建计算机系统任务（包括硬件和软件组件）的数学方法[26]。

FM的起源可追溯到数学逻辑，更确切地说，可追溯到产生逻辑计算机科学（LICS）和理论计算机科学（TCS）的数学逻辑。有关FM的起源，其中最著名的是艾伦图灵（Alan Turing）[42]进行的关键概念和理论方面具有里程碑意义的工作，特别是以一种精确术语来描述通常为计算机科学尤其是FM中核心算法的固有局限性。上述问题的可计算性已成为20世纪30年代自动计算领域科学发展的主要课题之一。这个时期科学讨论的核心是关于形式化系统中关键概念的解释说明，能够利用明确区别的语法和语义，以及形式化证明演算的概念来定义计算步骤[41]。LICS和TCS的一些领域有助于FM的基本开发，如语言和自动机（Automata）原理、编程/规范语言的语法和语义以及程序验证。最近，在系统规范和分析方面作出巨大贡献，特别是对于并发/分布式系统。这些贡献包括特定符号定义或符号分类、基础数学和/或支持这些符号的逻辑理论的发展、系统模型及其需求模型的自动或半自动分析有效算法与技术的发展以及实现上述算法可靠性自动工具的开发。接下来，简要介绍一些与基础理论，如进程代数，Petri网，类似于VDM、Z、B的基于状态的方法，时态逻辑，以及模型校验和定理证明等

分析技术的符号（符号类）最相关的示例。同时，还回顾了抽象解释，尽管没有与任何特定符号相关，但作为一种近似语义学的一般理论，对 FM 的发展也具有重要贡献。在此需要强调的是，限于本书篇幅，接下来的内容并不是要对 FM 进行全面综合分析，只是根据发展时间对该领域进行概述：

- 符号和相关支撑理论

进程代数。进程代数[30]是一种研究并发进程的代数方法。其工具是进程规范和形式化语句的代数语言，以及用于验证这些语句的演算定义同余定律。典型的进程代数运算符有顺序组合、不确定组合和并行组合。一些主要的进程代数有 CCS、CSP 和 ACP。

Petri 网。Petri 网最早是由 Petri[33]提出的，用以描述有限状态机的相互作用，并由一组位置有限集、迁移有限集、从位置到迁移以及从迁移到位置的关系流以及关系流相关权重组成。Petri 网可以形象地图形化表示，使得参数直观易懂。Petri 网的状态是通过标记位置而给定的，也就是说，在位置处关联一些令牌。定义规则通过移动令牌来触发迁移，从而改变网络状态。现已提出多种扩展和改进的 Petri 网，如增加令牌值[18]、增加时间[19,29]或增加迁移概率[3]等。

VDM、Z、B。第一个广泛使用的形式化规范语言，其采用传统数学概念，如集合、函数、一阶谓词逻辑等。VDM[5]的提出主要是用于描述程序设计语言中的指称语义。Z[38]采用相同概念来定义用于描述实体和研究系统状态空间的类型。在 Z 中，通过不变式谓词来描述状态空间的属性。状态迁移是通过模型运行的输入/输出（I/O）关系来表示的。B 方法[1]通过抽象机来增加行为规范。相应的程序语言通过一种基于细化的开发方法来补充实现，其中包括细化一致性的定理证明。文献［2］是对上述数学框架中系统建模关键概念一个很好的非正式介绍。

时态逻辑。时态逻辑是一种特殊类型的模态逻辑，提供了形式化系统的定性描述以及如何随着系统计算来推理声明的真值。典型的时态逻辑算子包括 *sometimes P*，其属性为未来某一时刻 *P* 为真，相应的，如果未来所有时刻 *P* 均为真，则称之为 *always P*。若模型应用时间和/或提供的时态算子具体设置不同，则特定的时态逻辑也不同（如线性时间和分支时间）。

- 分析技术和工具

模型校验。模型校验是一种采用高效算法以自动方式进行检测的检验技术，无论期望属性是否适用于一个（通常有限）系统模型，通常为自动机[4,7]的典型状态迁移结构。现已开发出功能强大的逻辑来表述各种系统性能，并对具体系统模型进行高级语言设计。前一个示例是各种时态逻辑的变形，后一个典型示例是进程代数、命令式语言和图形符号。模型校验器的显著示例是 SMV[9]、SPIN[22]和 TLC[28]。

自动机定理证明。自动机定理证明是指利用计算机来证明特定定理为真的过程。相关定理可能是传统数学领域的，也可能是数字计算机设计等其他领域的[37,44]。在用于系统验证时，系统规范 S 及其实现 R 是某些正确逻辑的公式。检验实现是否满足公式 $S \Rightarrow R$ 有效性的规格个数，这可由计算机程序（即定理证明器）自动实现，至少部分证明，有时完全证明。经典的定理证明器为 PVS[39]、HOL[43]和 Nqthm[6]。

抽象解释。抽象解释是一种（编程或规范）语言的语义近似理论。可用于数据和控制。该方法规范了语义精确性取决于观测程度的思想。如果近似程度足够粗糙，则语义抽象的精度较低但可用于计算。由于信息的相应缺失，并非所有问题都有答案，但经近似语义有效计算得到的所有答案都是正确的[12]。

FM 与 LICS 和 TCS 学科之间的主要区别是，前者作为一种思维工具试图为（软件或系统）工程师提供。

> 概念和技术来清晰、适当并方便地支持对复杂软件和硬件系统的描述、推理和构建[41]。

因此，重点在于构建而不是还原，同时着重于语用学而非完备性等经典问题。这种转变的重点是一般应用于 W. Thomas 称为的计算机科学逻辑[41]，而不是非常传统的计算机科学逻辑，当然尤其是要适用于 FM。那么在 FM 中就一定能成功吗？尽管尚不能给出一个完备答案，这是因为还没有看到 FM 在计算机工程工业中得到完全广泛的应用，但在判断是否正确回答上述问题上有非常明显的趋势，这将在下面简要阐述。

在过去，FM 已广泛应用于安全性、容错性、基本一致性、面向对象编程、编译正确性、协议开发、硬件检测、计算机辅助设计和人类安全（见文献 [32, 46] 中关于 FM 在上述领域中的大量应用）。在一些 IT 企业（如 IBM、Intel、Lucent/Cadence、Motorola 和 Simens 等主要的大型企业）的生产过程中采用 FM 技术来进行产品质量保证[10,31]，并从以下比尔·盖茨的语录中可清楚地看到 FM 正努力进入软件开发的实践领域：

> 像软件验证的工作，几十年来一直是计算机科学的崇高理想。现在在一些关键领域，例如在驱动程序验证方面，正在开发可用于真正验证软件及其如何工作以保证可靠性的工具[17]。

FM 及其相关工具也可作为基础科学分支中的支撑技术，如生物学（参见文献 [40]）、物理学和其他学科[24]。而且，FM 还可用于关键安全系统的设计和验证，如航空航天领域㊀[34]。在关键安全系统工程的国际标准中，如航空业或

㊀ 在欧洲形式化方法[16]的官方网站中可获得如何选择合适 FM 的操作指南。

运输业，越来越多地建议或强制采用 FM，如工程标准 ECSS-E-40-01[15] 和 CENELEC EN 50128[14]。在其他科学界也意识到 FM 的重要性，尤其是可靠性方面，例如，可靠系统与网络国际会议的可靠性软件密集型系统主题中专门组织模型校验研讨会，这是该领域最具影响力的会议（讨论在可靠性领域，FM 的重要作用以及未来更显著的作用，参见文献［45］）。后来，一些商业机构也提供面向 FM 的服务（文献［11］中所列部分）。但遗憾的是，关于 FM 是否已成功应用的问题仍然不能得到肯定回答，这是因为工业中的许多工具都是很少公开详细信息的专用工具。同样，由于类似原因也很难获得这些工具和方法的具体使用方法和过程㊀。不过，在文献［21］中介绍了关于 AT&T 贝尔实验室的 NewCoRe 项目，并明确表明 FM 在软件开发过程中的作用。

另一方面，通过科学/技术学科自身的发展进步也可评估该学科的成功。在过去几年内，在 FM 领域取得了巨大进展。首先，影响范围已经从单纯的行为功能方面扩展到如下的系统非功能性特征方面[27]：

- *空间移动性*。已扩展和开发了一些演算和逻辑以及相关支撑工具以显式处理如计算元素所占用（物理、离散）空间、从一个位置迁移到另一个位置以及蕴涵在网络连通性和通信结构中的概念。因此，在开发或推理大型分布式网络应用中重要的空间移动性基本概念，是这些形式化框架中的第一类对象。
- *安全性*。已扩展和开发了一些演算和逻辑以及相关支撑工具以用于安全协议及其期望属性的建模，以及验证或篡改前者满足后者。
- *有界资源*。开发基于逻辑、类型和演算的方法以及相关支撑工具用于资源分配和回收的建模和控制，并表示资源配置及传播中的信任度。
- *连续时间（基本的混合系统）*。在过去的 15 年，自动机、进程代数和逻辑以及相关支撑工具，在连续性方面进行扩展，如时间和连续变量导数函数等。
- *随机行为*。随机行为元素，如概率选择和随机时间，已引入到自动机和进程代数等模型中，并在正确逻辑中增加相应算子，从而提出形式化概念，为有关的性能与可靠性属性以及需求的建模和推理提供基于语言的工具。（进程代数/自动机）离散仿真以及马尔可夫链和马尔可夫决策过程模型校验的自动软件工具构成相应的实践支撑部分。

在文献［31］中列出了上述领域的发展以及大量相关参考文献。此外，也扩大了 FM 的应用领域，包括新的学科，如计算生物学。而且，FM 支撑工具的功能也显著提高，例如一些模型校验技术。

㊀ 某些 FM 倡导者认为这本身就是对 FM 及其相关工具在工业策略中的一种证明，其本身就可表明在工业策略中的成功。

在一些难以利用传统方法调试和测试的并发软件中应用效果良好[22]，如今，模型校验器可以处理多达 10^{100} 个状态的系统模型，某些情况下，甚至可以处理超过 10^{100} 个状态的系统模型[4,20]。然而，在 FM 领域仍然存在一些未解决的问题，如：

- 严格规范的技术不易扩展，尤其是自动校验技术。这是由于状态爆炸问题使得在系统建模和验证时会并行发生许多独立行为。

- 尽管在并发系统的组成规范领域具有显著进步（特别是引入进程代数），但在系统模型验证中却无太大改善。特别是在目前的模型校验技术中，尚未充分利用组合性。

- 许多规范范式和验证技术各自独立开发，往往造成技术上的不当二分法。不同范式之间缺乏充分整合，如一些规范语言中面向状态的范式与其他规范语言中面向行为或面向对象的范式，或不同验证技术之间，如自动定理证明与模型校验，都可能不利于每种范式或技术。

- FM 应尽量在系统中完全接受并成功应用，尤其是软件系统工程，并应能够顺利地嵌入到更传统的工业生产过程中。

1.2　未来发展方向

在中/长期的未来中，FM 面临的严峻挑战将是大规模的工业化和加强基础研究工作，对于解决计算机可靠性，特别是软件可靠性，这是必须的。软件可靠性将是计算机科学和实践中的重大挑战[11]。

从更具技术的层次上，需要特别在一些研究方向上着重发展。接下来，简要介绍了其中几个研究方向。尽管下列方向并不全面，但必然具有一定关联性；所列方向的顺序并不意味着其优先级或重要程度：

- 在抽象解释方面，“已开发出通用的、富有表现力且成本效益较高的抽象来处理浮点数、数据依赖性（如并行化）、公平生命周期性 [⋯]、嵌入式软件的时间特性和概率特性等。应增强目前的工具，用于处理高阶组合模块分析，并处理涉及复杂数据和控制概念的（如对象、并发线程、分布式/移动编程等）新的编程范式，尤其是自动合并和局部细化抽象以处理‘未知’答案”[11]。此外，还应探索用于（扩展）静态分析的抽象新方法，其中包括采用定理证明[35]。

- 在模型校验方面[7,8,24]，抽象概念是一项主要内容。将在特别是克服状态爆炸问题方面具有重要作用。另外，还允许将模型校验扩展到无限模型。最后，可实现软件程序的模型校验而不仅是规范研究[23]。需要研究将抽象技术嵌入到模型校验中。以下是一些需要进一步研究的问题：①如何从实际中自动

构建抽象模型；②当检测到抽象模型报告出错时，即抽象模型包含的信息比实际模型少，如何通过模型校验算法来辨识虚假的抽象反例；③如何细化抽象模型以消除检测到的虚假反例；④如何从抽象（非虚假）模型中推导出具体反例；⑤如何开发验证参数化系统的方法，即系统中具有任意多的相同组件。另一个主要方法是组合性。不可能对未来复杂的真实系统进行模型校验，即一般意义上的分析，如无法研究其结构和架构的全局普适系统，也就是说，该系统由简单的组件组成。组合推理和组合模型校验在并发系统的自动检测领域具有重要作用。最后，模型校验和定理证明的顺利整合将会对示例验证产生巨大飞跃，便于无限状态系统的建模和检验，并通过基本情况的有效证明得到功能强大的归纳原则。

● 在 FM 的安全性、资源获取和信任度领域，期望开发出面向安全的程序语言，并直接来自于 1.1 节中提到的移动性和安全性的理论和演算。除此之外，还需开发移动智能体的特定协议来获取和管理资源，这将把访问协商放在信任评估之上，且信任程度取决于智能体的相关知识和信念，以及如何传播信任度。并严格调查和评估该协议，包括自动验证[27]。

● 在混合系统领域，进一步开发现有工具中通过混合自动机（包括定时自动机）中的复杂连续函数（包括定时自动装置）来生成状态集的基本思想，并开发相关的高效算法，以有利于混合系统模型的自动推理。其中主要工作在于加强与控制理论的统一[27]。

● 进一步开发随机概率模型的模型校验技术和逻辑，以扩展到现实生活系统中。重点在于研究反例产生问题，以及数值分析算法和各种模型校验所需的技术。而且，为了建模和验证未来全局普适计算算法的重要性，需要迁移模型和上述混合模型[27]。

● 博弈语义是指一种形式化规范/编程语言和逻辑语义的一种很有发展前景的方法，该方法是从数学竞赛、逻辑、组合博弈理论和范畴论发展而来的。尽管该方法奠定了 FM 的多个基础，另外，还将为模型校验（尤其是在组合性方面）、部分规范和建模的发展提供坚实基础，并集成定性和定量方法，开发基于物理的计算模型，如量子计算[27]。

● 最后，为正确面对 FM 的“工程挑战”，必须设计一种公式和图表适当结合的语言。目前，已在软件工程领域，特别是工业方面，进行了最显著的 UML 尝试，尽管从数学基础的观点看来，并没有特别令人印象深刻。另一方面，还进行了其他一些尝试，历史上的例子如布尔表达式的等价性和有序二元决策表，或正则表达式和有限自动机，这些都已相当成功并产生非常有用的工程工具。这将推动“支持图表语言和公式语言合并的理论”的研究，因为这“有助于设计更好地规范语言”[41]。

尽管上述列举远不完整，但仍可认为以上所有行为对于正确面对信息系统[25]和 TCS 及其应用[36]的巨大挑战，并在计算机科学和工程以及其他分支科学和工程中挖掘 LICS 和 TCS 的潜力[24]是必不可少的。

致谢

在此，作者衷心感谢 Peter Neumann（SRI 国际，美国）、Rocco de Nicola（佛罗伦萨大学，意大利）和 Scott Smolka（SUNY，美国）提出的宝贵意见；感谢 Stefania Gnesi 和 Alessandro Fantechi 在很多问题上的无私帮助；感谢 Mieke Massink 和 Angelo Morzenti 审阅本章内容。

参考文献

1. J. R. Abrial. *The B-Book*. Cambridge University Press, 1996.
2. J. R. Abrial. Faultless systems: Yes we can. *IEEE Computer Society*, 42(9):30–36, 2009.
3. M. Ajmone Marsan, G. Conte, and G. Balbo. A class of generalized stochastic Petri nets for the performance evaluation of multiprocessor systems. *ACM Transactions on Computer Systems*, ACM Press, 2(2):93–122, 1984.
4. C. Baier and J.-P. Katoen. *Principles of Model Checking*. The MIT Press, 2008.
5. D. Bjorner and C. Jones. *Formal Specification and Software Development*. Prentice Hall International, 1982.
6. R. Boyer. Nqthm, the Boyer-Moore prover, 2003. Available at: http://www.cs.utexas.edu/users/boyer/ftp/nqthm/index.html.
7. E. Clarke, E. Emerson, and J. Sifakis. Model checking: Algorithmic verification and debugging. Turing lecture. *Communications of the ACM*, 52(11):75–84, 2009.
8. E. Clarke, O. Grumberg, S. Jha, Y. Lu, and H. Veith. Progress in the state explosion problem in model checking. In R. Wilhelm, ed., *Informatics 10 Years Back 10 Years Ahead, Volume 2000 of Lectures Notes in Computer Science*, pp. 176–194. Springer-Verlag, 2000.
9. E. Clarke, O. Grumberg, and D. Peled. *Model Checking*. MIT Press, 1999.
10. E. Clarke, J. Wing, et al. Formal methods: State of the art and future directions. *ACM Computing Surveys*, ACM Press, 28(4):626–643, 1996.
11. P. Cousot. Abstract interpretation based formal methods and future challenges. In R. Wilhelm, ed., *Informatics 10 Years Back 10 Years Ahead, Volume 2000 of Lectures Notes in Computer Science*, pp. 138–156, 2000.
12. P. Cousot. Abstract Interpretation and Semantics, September 30, 2003. Available at: http://www.di.ens.fr/~cousot/Equipeabsint-eg.shtml.
13. A. Emerson. Temporal and modal logics. In J. van Leeuwen, ed., *Handbook of Theoretical Computer Science—Vol. B: Formal Models and Semantics*, pp. 995–1072. Elsevier, 1990.
14. European Committee for Electrotechnical Standardization. CENELEC. Railway application—Communications, signalling and processing systems—Software for railway control and protection systems. CENELEC EN 50128. 2011.

15. European Cooperation for Space Standardization ECSS. Space segment software. ECSS E-40-01. 1999.
16. Formal Methods Europe. FME Home Page, September 30, 2003. Available at: http://www.fmeurope.org.
17. B. Gates. Remarks by Bill Gates. WinHEC 2002 Seattle—Washington April 18, 2002. Available at: http://www.microsoft.com/billgates/speeches/2002/04-18winhec.asp [Accessed October 6, 2003].
18. H. Genrich. Predicate/transition nets. In G. Rozenberg, ed., *Advances in Petri Nets 1986, Volume 254 of Lectures Notes in Computer Science*, pp. 207–247. Springer-Verlag, 1986.
19. C. Ghezzi, D. Mandrioli, S. Morasca, and M. Pezzè. A unified high-level Petri net formalism for time-critical systems. *IEEE Transactions on Software Engineering*, 17(2):160–172, 1991.
20. L. Hoffman. Talking model-checking technology. A conversation with the 2007 ACM A. M. Turing Award winners. *Communications of the ACM*, 51(7):110–112, 2008.
21. G. Holzmann. The theory and practice of a formal method: NewCoRe. In *Proceedings of the 13th IFIP World Congress*, pp. 35–44. IFIP, 1994.
22. G. Holzmann. *The SPIN Model Checker. Primer and Reference Manual*. Addison-Wesley, 2003.
23. R. Jhala and R. Majumdar. Software model checking. *ACM Computing Surveys*, 41(4):21:2–21:54, 2009.
24. D. Johnson. Challenges for Theoretical Computer Science, 2000. Draft Report from the Workshop on Challenges for Theoretical Computer Science held in Portland on May 19, 2000. Available at: http://www2.research.att.com/~dsj/nsflist.html.
25. A. Jones, ed. Grand research challenges in information systems. Computer Research Association, 2003.
26. C. Jones. Thinking tools for the future of computing science. In R. Wilhelm, ed., *Informatics 10 Years Back 10 Years Ahead, Volume 2000 of Lectures Notes in Computer Science*, pp. 112–130, 2000.
27. M. Kwiatkowska and V. Sassone (moderators). Science for Global Ubiquitous Computing, 2003. Proposal for discussion nr. 2 in the context of the Grand Challenges for Computing Research initiative sponsored by the UK Computing Research Committee with support from EPSRC and NeSC.
28. L. Lamport. *Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers*. Addison-Wesley, 2002.
29. P. Merlin and D. Farber. Recoverability of communication protocols. *IEEE Transactions on Communications*, 24(9):1036–1043, 1976.
30. R. Milner. Operational and algebraic semantics of concurrent processes. In J. van Leeuwen, ed., *Handbook of Theoretical Computer Science—Vol. B: Formal Models and Semantics*, pp. 1201–1242. Elsevier, 1990.
31. R. Milner, A. Gordon, V. Sassone, P. Buneman, F. Gardner, S. Abramsky, and M. Kwiatkowska. Theories for ubiquitous processes and data. Platform for a 15-year grand challenge, 2003. This paper is written as a background for Reference [27].
32. P. Neumann. Practical architectures for survivable systems and networks. Technical Report Cont. 1-732-427-5099—Final Report, SRI International, 2000. Available at:

http://www.csl.sri.com/users/neumann/ [Accessed September 30, 2003].

33. W. Reisig. *Petri Nets—An Introduction, Volume 4 of EATCS Monographs on Theoretical Computer Science*. Springer-Verlag, 1985.
34. J. Rushby. Formal methods and the certification of critical systems. Technical Report CSL-93-7, SRI International, 1993. Also issues under the title *Formal Methods and Digital Systems Validation for Airborne Systems* as NASA CR 4551.
35. K. Rustan and M. Leino. Extended static checking: a ten-year perspective. In R. Wilhelm, ed., *Informatics 10 Years Back 10 Years Ahead, Volume 2000 of Lectures Notes in Computer Science*, pp. 157–175, 2000.
36. A. Selman. Challenges for Theory of Computing, 1999. Report of an NSF-Sponsored Workshop on Research in Theoretical Computer Science held in Chicago on March 11–12, 1999. Available at: http://http://www.cse.buffalo.edu/~selman/report/.
37. N. Shankar. Automated deduction for verification. *ACM Computing Surveys*, 41(4): 20:2–20:56, 2009.
38. J. Spivey. *The Z Notation—A Reference Manual*. Prentice Hall International, 1989.
39. SRI International—Computer Science Laboratory. The PVS Specification and Verification System, September 30, 2003. Available at: http://pvs.csl.sri.com/.
40. The BioSPI Project. The BioSPI Project Home Page, October 27, 2003. Available at: http:// www.wisdom.weizmann.ac.il/~biopsi.
41. W. Thomas. Logic for computer science: The engineering challenge. In R. Wilhelm, ed., *Informatics 10 Years Back 10 Years Ahead, Volume 2000 of Lectures Notes in Computer Science*, pp. 257–267, 2000.
42. A. M. Turing. On computable numbers with an application to the entscheidungsproblem. *Proceedings of the London Mathematical Society*, 2(42):230–265, 1936.
43. University of Cambridge—Computer Laboratory. Automated Reasoning Group HOL page. Available at: http://www.cl.cam.ac.uk/research/hvg/HOL/.
44. Various Authors. Wikipedia, September 30, 2003. Available at: http://www.wikipedia.org
45. J. Woodcock (moderator). Dependable Systems Evolution. A Grand Challenge for Computer Science, 2003. Proposal for discussion nr. 6 in the context of the Grand Challenges for Computing Research initiative sponsored by the UK Computing Research Committee with support from EPSRC and NeSC.
46. J. Woodcock, P. G. Larsen, J. Bicarregui, and J. Fitzgerald. Formal methods: Practice and experience. *ACM Computing Surveys*, 41(4):19:2–19:36, 2009.

第2部分

建 模 范 例

第2章　一种正在应用的同步语言：LUSTRE 的发展

Nicolas Halbwachs
Vérimag，格勒诺布尔，法国

2.1　简介

同步语言 LUSTRE 的设计始于 20 多年前，并相应开发了一种工业软件开发工具 SCADE，目前应用于很多大型企业中的嵌入式软件开发（航空电子设备、交通运输、能源等）。这似乎是非常值得报道和分析的一个成功案例，并从一项新技术到产业转移的角度考虑：为什么会成功？如何才能更加成功？

相对于首次应用于大型工业的 20 世纪 80 年代，现在，嵌入式系统更加流行。此时需承认这一领域涉及的系统具有一个或多个以下特征：①必须在与环境，可能是物理环境（实时系统、工业控制等），紧密相互作用下运行；②系统的发展综合了软件和硬件方面；③系统受到严格的非功能性限制（执行时间、内存限制、容错性、功耗等）；④由于①中的环境对工艺过程和设备的影响，系统的关键是必须安全。由于①和②，嵌入式系统与控制理论和硬件设计有着紧密联系，这也是为何同步语言的灵感来自于控制工程形式化和硬件描述语言的原因。

或许，大家可能非常想知道为什么 3 种主要的同步语言 ESTEREL[4]、SIGNAL[27] 和 LUSTRE[18] 都源自于法国，且几乎是在同一时间独立发展起来的。当然，在 20 世纪 80 年代初，同步思想就已出现，不管是 Milner[30] 的理论研究，还是 Grafcet（或 IEC 1131 顺序功能图）[3,13] 或状态图[23] 的“几乎同步”形式化。但无论从学术界方面还是从工业界方面来看，条件都是特别有利的。

在学术方面，有包括从事控制理论和计算机科学的研究人员组成的 3 个团队：分别是 Jean-Paul Rigault、Jean-Paul Marmorat 和 Gérard Berry 研究的 ESTEREL；Albert Benveniste、Paul Le Guernic 研究的 SIGNAL；Paul Caspi 和本章作者研究的 LUSTRE。这两种能力的结合在语言设计中具有重要作用。

另一方面，社会上面临着强烈的工业需求：欧洲和法国的嵌入式软件行业面临着巨大挑战：

- 法国核反应堆首次诞生称为 N4 的新家族，最主要的功能（尤其是紧急

停机）是由计算机系统 SPIN（用于集成核保护系统）实现。

- 与此同时，第一架完全“应用全数字电传操纵”的飞机——空中巴士 A320 设计诞生。
- 在铁路行业，设计出各种自主运行的地铁（如 VAL[15]）以及法国 TGV（高铁）系列也越来越完全由计算机运行。

从 1984 年开始，得益于非常好的市场环境，LUSTRE 得到快速发展。首先在简要回顾同步语言的原则（2.2 节）之后，将在 2.3 节从学术和工业角度详细介绍同步语言发展的主要阶段。2.4 节分析了同步语言在工业应用中的情况。最后，2.5 节概述了同步语言的目前进展及相关工具。

本章的最初内容发表在文献［16］中。

2.2 同步语言风格

首先以一种简单方式来回顾 LUSTRE 的原理：通过数据流来执行 LUSTRE 程序。任一变量（或表达式）x 表示一个数据流，即一个无限序列（x_0，x_1，…，x_n，…）。设计一个具有周期行为的程序，其中 x_n 为执行到第 n 个周期的 x 值。程序由输入流计算得到输出流。通过等式（数学意义上）运算得到输出流（可能为局部），等式“x = e”意味着“$\forall n$，$x_n = e_n$”。因此，一个等式可认为是一个时间不变量。LUSTRE 运算符对数据流进行全局运算：例如，“x + y”表示数据流（x_0+y_0，x_1+y_1，…，x_n+y_n，…）。除了常用的算术运算，还可以如数列所示逐点进行布尔运算和条件运算，在此仅考虑两种时态算子：

- 运算符“pre”（“之前”）允许访问变量的先前值：“pre(x)”表示数据流（nil，x_0，…，x_{n-1}，…），其中，首个值 nil 为一个未定义（“未初始化”）值。
- 运算符“- >”（“之后”）用于定义初始值：“x- >y”表示数据流（x_0，y_1，…，y_n，…），初始时等于 x，然后就总等于 y。

一个简单而经典的示例是如下所示的“事件”计数器程序：两个布尔数据流“evt”（只要计数“事件”发生则为真）和“reset”（只要需重新初始化计数器则为真）作为输入，返回最近一次“复位”后“事件”发生的次数：

```
node Count (evt, reset: bool) returns (count: int);
Let
  count = if (true- > reset) then 0
    else if evt then pre (count) + 1
    else pre (count);
tel
```

可直观地看出，“true- > reset”是一个布尔数据流，在初始时刻和“reset”为真时为真；即若“true- > reset”为真，则“count”值为0；若“event”为真，“count”值加1。否则，保持初始值。

一旦声明，该“node”可作为一个用户定义运算符用于程序中的任何地方。例如，该计数器可通过计数“second”值并对 60 取模，每隔 60“second”产生事件“minute”：

```
mod60 = Count （second, pre （mod60）=59）;
minute = (mod60 =0);
```

其中，“mod60”是“Count”节点的输出，计数“second”，并当“mod60”的数值为 59 时复位，若“mod60”为 0，则“minute”为真。

因此，通过节点概念，LUSTRE 可提供层次化描述和组件复用。沿运算符网络“线路”传输的数据可以是复杂的结构体信息。

从时序角度来看，工业应用表明在单个系统中可出现以不同速率生成的多个进程链。在 LUSTRE 中提出一个布尔时钟概念，使得允许以不同速率激活节点。

LUSTRE 文本语法的相应图形化表示非常直观，例如：图 2.1 是 SCADE 中显示的上述“minute detector”。

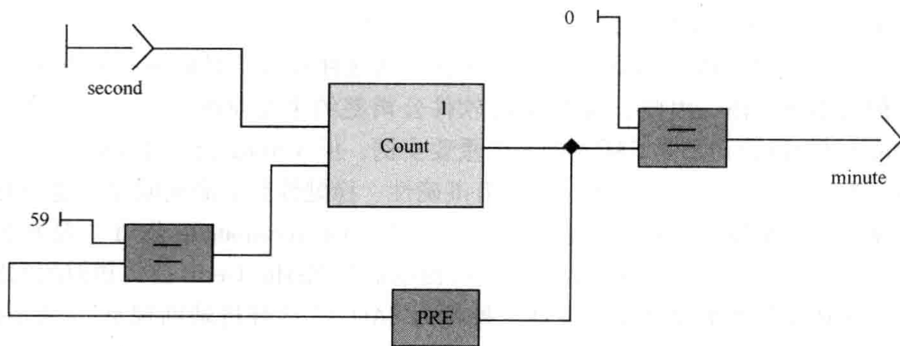


图 2.1　SCADE 的图形化视图

2.3 LUSTRE 和 SCADE 的设计和开发

LUSTRE 的初始想法源自之前通过时间函数对实时系统建模的研究工作[9]，根据这种变量“历史”的全局处理以及 Lucid⊖同步语言的启发，设计出一种将

⊖ 实际上，“LUSTRE”一词是“实时的同步 Lucid”的法文缩略语。

时间序列值描述为变量的编程语言。而且，在控制理论背景下，Paul Caspi 认为对于控制工程人员，这种描述方式是非常自然的，现在将该方式称为“MATLAB/Simulink 模式”；根据以前的技术经验，利用该语言来声明或形式化数据流，即高层的微分方程或有限差分方程，以及底层的各种图结构（框图、模拟网络、开关设计图等）。

2.3.1 工业发展

Caspi 的直觉很快得到确认：目前在很多公司，许多内部开发工具都是基于这种形式化语言的。这种工具的用途广泛，包括从简单的图形描述（无需任何机械开发思想）到进行一致性验证、模拟仿真，甚至是自动代码生成。尤其是，位于格勒诺布尔的 Merlin Gerin 公司（即现在的施耐德电气公司）所负责的大部分 SPIN（“集成核反应保护系统”）开发。意识到由于软件的缺陷所面临的全新问题，管理人员决定研究自己的开发环境，并很自然地选择了形式化数据流。在此，一个良好机遇是从一个称为 SAGA（“辅助规范和自动生成”）开发环境的初始设计阶段进行合作的。基于 LUSTRE，SAGA 具有图形/文本的混合语法，并提供一种简单而高效的代码生成器。LUSTRE 团队的两个成员：Eric Pilaud 和 Jean-Louis Bergerand 被 Merlin Gerin 雇佣来监督该工具的开发。SAGA 成功应用于 SPIN 和其他多个系统的设计开发。

但几年之后，Merlin Gerin 公司人员意识到这种软件工具的维护和开发并不是他们的本职工作。因此，与 Verilog 软件公司签约来商业推广 SAGA 产品。尽管 Merlin Gerin 公司已是 SAGA 的一个重要实例，但 Verilog 公司还是接受了这一挑战：对于一个小公司，开发一种具有准确性、稳健性和生命周期等关键约束的工具是一种全新挑战。Verilog 公司与位于图卢兹的 Aerospatiale 公司［现为空中客车（Airbus）子公司］签约，为解决所面临的与 Merlin Gerin 公司相似的问题，在基于 SAGA 的类似原理上，设计一种称为 SAO（“计算机辅助规范”）的内部工具以用于空中客车 A320 机载软件的设计开发。但起初的 SAO 并不打算实现嵌入式代码的自动生成。后来，Aerospatiale、Merlin Gerin 和 Verilog 公司之间建立了一种联盟，在 SAO 和 SAGA 的启发下设计出一种称为 SCADE（用于“安全关键应用开发环境”）的新工具。与此同时，Verilog 公司和我们的学术机构共同建立公共实验室 VERIMAG，以更加便于 SCADE 的合作设计开发。Verilog 雇用 LUSTRE 团队成员 Daniel Pilaud 来负责 SCADE 团队。

SCADE 面临的新的巨大挑战是需要符合航空电子设备认证机构的要求。尤其是，航空电子设备标准 DO-178B 要求用于关键设备开发的任何工具必须是与研究设备具有相同关键程度下自身合格的。因此，为实现 SCADE 产生的代码是可嵌入的，SCADE 代码生成器应在与关键软件相同等级（飞行软件，A 级）下

质量合格。尽管认为这种资质与编译器的形式化证明无关，但这仍是一种设计过程，包括测试范围、程序文件质量、可追溯性需求等问题。SCADE 代码生成器 KCG 或许是第一个能够用于验证民用航空电子设备嵌入式软件的商业化编译器。KCG 是 SCADE 的一个重要参数：作为一种任何代码的生成器，不仅受限于人工代码的数据流规范，更重要的是，由于质量保证的需要，还应进行成本很高的一致性测试来检测编译正确性。这就是所谓的从“V”循环到“Y”循环的变化过程：由代码和单元测试构成的“V”的最底层部分是免费且瞬时的。

在 20 世纪 90 年代，瑞典公司 Prover Technology 将基于 SAT 的模型校验器 PROVER 和 SCADE 相结合，使之具有集成验证功能。为此，采用形式化技术来指定属性并通过同步观测器进行假设。

接下来，同步语言 LUSTRE 的工业化过程变得复杂：首先，Verilog 公司被 CS 集团收购，然后卖给 Telelogic 公司，该公司又将 SCADE 技术卖给创建于 1999 年主要以 ESTEREL 语言开发工业工具的 Esterel Technologies 公司。ESTEREL 语言的主要应用领域是电路 CAD 设计，而 SCADE 是嵌入式软件，因此这两种语言并不相互对立。尽管在外部结构上相当不同，但都具有相同的同步语义模型。因此同时利用这两种工具共同开发是很有意义的，甚至可以将两者合并（见 2.5 节）。Esterel Technologies 公司购买 SCADE 后，重新恢复 SCADE 的开发（尤其是在项目 IST-RISE 框架下开发的 Simulink/Stateflow[7] 的接口）以及商业宣传推广：现在全世界范围内都在使用 SCADE。

2.3.2 研究阶段

LUSTRE 及其相关工具的研究是在与致力于其他同步语言的并在称为 C2A 的法国和欧盟项目 Eureka-SYNCHRON 和 Esprit/LTR-SYRN㊀以及 SafeAir-Ⅱ㊁中成功应用的研究团队竞争/合作的激烈背景下进行的。这项研究还涌现出一系列学位论文，但遗憾的是，这些论文都是法文撰写的。

2.3.2.1 编译

在同步语言设计之后的第一个研究课题（Jean-Louis Bergerand 的学位论文）理所应当是同步语言的编译。LUSTRE 可以很自然地编译成顺序命令代码。以下是一个无限循环示例：

```
initializations
loop
```

㊀ 参见 http://www-verimag.imag.fr/SYNCHRONE/SYRF/syrf.html。

㊁ 参见 http://www.safeair2.org/。

```
    acquire inputs;
    compute outputs;
    update memories
end
```

为最小化存储单元个数（通常，无需两个存储单元来处理 x 和 pre(x)），只需确定一个计算输出结果和存储单元的正确顺序。

然而，在 20 世纪 80 年代中期，编译 ESTEREL 的唯一方法是产生一个显式自动机。在此，采用一种类似于 LUSTRE 编译的方法（John Plaice 的学位论文）。该方法的思想是通过布尔表达式已知的*之前*值来规范代码：某一循环中“b”为真时，那么在下个循环中“pre(b)”将为真，而且可相应地简化该代码。当然，也可通过运算符“- >”的语义来指定初始循环。因此，自动机的状态对应于布尔变量存储器的配置，且根据状态来指定的代码表示输出迁移、运行条件和作用于非布尔变量的行为。与 ESTEREL 团队合作，定义一种通用格式 OC[34] 作为自动生成器的目标代码，以便在这种格式下共同常用工具，如各种目标语言（C、Ada 等）的代码生成器、最小编译器、最优编译器和图形可视化工具。

与 ESTEREL 相比，该代码生成器可及时产生具有冗余状态的自动机，因此甚至在简单程序中也可能出现代码爆炸。这就是为何提出第一种“符号互模拟”算法[6,22]的原因，目的是在自动机生成代码时执行互模拟还原。通过二元决策图（BDD）技术实现的这种优化方法是 Pascal Raymond 学位论文的主要内容。

显式自动机代码比单循环代码的效率高很多，但通常也大得多，而且代码大小是比其效率更重要的一种指标。自动机大小甚至与源程序大小成指数关系，因此显式自动机几乎不能用于实际大小的程序中。然而，产生控制自动机的各种方法对验证工具的设计非常有用。ESTEREL 和生成自动机的另一个影响是在语言中引入*断言*(assertions)：ESTEREL 中，在输入信号之间引入简单的排斥或暗示关系，使得编译器相应地简化自动机。在 LUSTRE 中，自然地推广“assert < expression >”的关系，来表明任意布尔表达式的不变性。

2.3.2.2 规范和验证

LUSTRE 程序验证是 Anne-Cécile Glory 和 Christophe Ratel 学位论文的主要研究内容。一旦已有一个可控自动机发生器，同时也具有了用于布尔程序的自动验证工具。例如，为了验证仅包含布尔变量和运算符的两个节点行为相同，如图 2.2 所示，只需将这两个节点连接，将生成程序编译到一个显式自动机中，并检查生成代码是否输出“ok”时从未赋值为“false”。这一操作也适用于以一种保守方式进行一般节点的比较：如果“ok”从未赋值为“false”，则这两个节点等价，否则该验证具有不确定性。一种更一般的验证方案如图 2.3 所示：一般来说，想要证明只要环境行为得当（即满足某种*假设*条件），程序就满足某些*性*

能。现在，如果仅限于验证*安全性能*，假设和性能都能够由称为*同步观测器*[20]的程序表示，即将所有相关变量作为输入，计算得到一个布尔变量输出，只要输入满足相应的性能或假设，则输出为真。将需要验证的节点连接到图 2.3 所示的观测器，验证过程包括表明（通过可控自动机上的模型校验）如果“env_ok”之前设置为“false”，则输出“prog_ok”只能设置为“false”。当然，由于状态爆炸，最好使用专用验证工具而不是编译器。一种称为 LESAR 的专用模型校验器已开发用于 LUSTRE[19]。该工具采用自动机枚举搜索技术或象征前进/后退技术进行校验。虽然该模型校验技术很经典，但在同步声明语言中非常自然地引入了观测器规范。在将工业验证工具与 SCADE 集成时，Prover Technology 公司也采用了相同的技术。另外，本团队还开发了一种称为 NBAC 的验证工具[21,25]，该工具基于线性关系分析，并能够处理数值变量的简单（线性）行为（Bertrand Jeannet 的学位论文）。

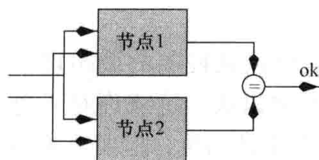


图 2.2　两个节点的比较

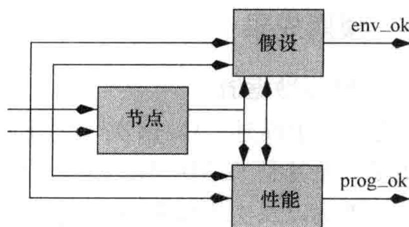


图 2.3　观测器规范

2.3.2.3　自动检测

采用与观测器规范相同的技术来执行自动检测[14,24,33,35]：假定观测器只用于产生实际的检测序列，而性能作为一种“标准”来确定每种测试通过与否。

2.3.2.4　硬件描述和数组

1989～1992 年，在与巴黎数字设备研究实验室合作过程中，采用 LUSTRE 来配置当时出现的大规模 FPGA（Frédéric Rocheteau 的学位论文）。为此，在 LUSTRE 语言中扩展了数组机制（显然，需要表示寄存器和常用结构），并出现在该语言的第 4 版中[36]。

2.3.2.5　分布式代码生成：动态规划

从同步程序到分布式构架的代码生成是一个具有挑战性的课题。主要解决用户自行分配时的问题，例如，为每个变量分配一个计算节点。在 Alain Girault 的论文[8]中提出了第一种解决方案，具体如下：①为每个节点复制整个序列代码；②根据局部计算的变量，在每个节点处进行代码分层；③假定一种简单通信机制（固定大小的 FIFO），增加通信代码及同步代码。在 Rym Salem 的论文[10]中提出了另一种受控制理论和工业应用启发的解决方案：该方法并不能够准确实现集中

式程序中的同步语义，但充分利用了同步和异步接口之间的不确定性（输入采样大多异步）。近年来，解决了真正多循环编程问题：通过*时钟*机制，LUSTRE允许变量以不同速率变化；然而，即使某个变量变化较“慢”，但其计算也必须在与其他变量相同的较“短”周期内完成。文献［37］提出一种生成真正多循环代码的方式，其中，*在严格遵守同步语义的条件下*，缓慢执行的任务被快速紧迫的任务抢占。

2.4 工业应用案例

现在，对10多年来LUSTRE/SCADE在工业中的应用情况做一汇报。其中，一些预期成果得到证实，而也有一些没有得到验证。同时也出现了一些意想不到的品质以及一些意料之外的需求。

2.4.1 预期成果

2.4.1.1 同步数据流

首先，如上所述，Caspi关于便于控制工程人员形式化控制的最初理念得到充分证实，尤其是SCADE用户非常自然地接受图形化语法。这或许对于更喜欢利用文本语法以及文本编辑器功能的程序员来说显得非常陌生，但要成为系统设计师，则必须采用图形化语法。事实上，图形化语法SYNCCHARTS[1]已在ESTEREL语言中得到定义。

2.4.1.2 形式化语义

事实上，与很多控制工程的形式相比，包含一种简单明了、*抽象*的形式化语义的LUSTRE具有很大的潜在优势：

- 可能会提高程序质量，这是因为提高了用户对编程语言的理解程度，然而难以衡量这种影响程度。
- 易于对程序形式化推理，但目前在工业领域，尚不清楚形式化推理是否是一项重要内容。
- 然而，对易于编译和提高生成代码质量具有巨大影响。如Grafcet、Statecharts或Simulink等编程语言都是通过仿真算法来定义的，这对于代码生成和优化没有太大的自由空间。
- 希望通过采用形式化语义，能够正式证明LUSTRE编译器的优点。虽然Eduardo Gimenez已成功尝试过Coq的辅助证明[5]，但工业编译器从未真正得到证明。主要问题可能在于目前的这种证明未被认证机构接受。

2.4.1.3 从时钟脉冲到激活条件

在LUSTRE中，时钟概念与在SIGNAL中具有重要作用的时钟概念非常类

似：允许一个数据流仅在某些周期具有数值，然后仅在这些周期触发运算符，SCADE 用户很少采用这种非常抽象的概念，认为这太复杂。因此，由一个称为“激活条件”的简单原语替代：一个布尔数据流可作为激活条件附加到某一节点：只有当条件为真时，节点才能激活，否则输出保持为初始值（而不是时钟机制中的“无”）。

2.4.1.4　形式化验证和自动检测

随着时间的推移，对于形式化验证结论的关心有所减轻。目前已知一些利用工业证明器插件工具或本团队技术原型进行的性能验证过程。其中，观测器技术更受欢迎，这是因为其采用同样的编程语言来编制程序和性能。然而，形式化验证并不是软件设计中的一项常规任务，甚至要求用户表示高级属性和假设都比较困难。此外，验证所产生的效应也难以量化：用户经常会否定是通过验证来提高最终程序可靠性的（不管怎样，本团队的程序是零错误的），而且在降低成本方面，也不承认验证可减少代码认证所需的测试量。在此，希望将自动测试看作一种“特洛伊木马”：因为自动测试已广泛接受（由于能够使现有工作更加容易），同时自动测试（即编写假设和性能的观测器）所需工作量基本上和验证所需的工作量相同，用户逐步会习惯于验证工作。

2.4.2　意外功能和需求

另一方面，还发现与现有工具相比，LUSTRE/SCADE 所具有的之前被忽视的其他优势，主要是因为对于现代编程语言，这是非常经典的：

- *程序架构*。与许多控制工程语言相比，通过节点概念，LUSTRE 能够通过少数预定义运算符构建任意深度的程序结构。像 SAO 这样的工具只能提供一个分区的程序表，而不可能实现嵌套，否则该工具基于一个庞大（而且仍在不断增加）的预定义运算符库。
- *编译效率*。与之前的工具相比，期望主要在于提高代码效率，事实上，某些用户已对该编译效率留下深刻印象，相比之下，一些现有的工具会耗时数小时来编译一个非常简单的程序。
- *即时循环检测*。在 LUSTRE 中，禁止变量一开始就取决于自身（循环中无“pre”）：这称为*因果关系错误*。值得注意的是，编译器对这种即时循环的检测可能会突出规范不一致性，当以一种顺序命令语言编程时会掩盖该不一致性：由于这种语言要求用户必须指定评估顺序，存在于规范中的一些问题循环可能会被忽视，这是因为受顺序语句影响，这些循环可能会在任意点处中止。

2.4.2.1　因果关系问题：模块化和独立编译

如上所述，同步语言可产生因果关系错误的问题（瞬时自相关性）。因果关系错误的精确检测是不确定的，这是因为其取决于任意的数据属性。因此需要近

似化处理：编译器可能拒绝执行一些可执行程序。在 LUSTRE 中，由于只是进行语法检测，因此近似程度更加粗略。然而，似乎这种粗略检测并不受用户干扰，甚至更加复杂：在 SCADE 中，准则甚至更严格㊀，因为禁止一个节点的输入瞬间取决于该节点的输出（即循环中不仅需要一个“pre”，并且还应出位于节点外；见图 2.4）。这种限制产生很大影响：节点可独立编译（与所有其他同步语言相比），这是因为节点内部的计算顺序并不取决于其调用方式。当然，独立编译是实际应用中一个重要课题。另外，对于认证时通常所需要的代码可追溯性也很重要（即关联产生代码与源程序的能力）：节点代码生成或节点调用都明确定义为整个目标代码中的连续块。

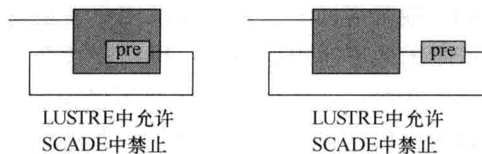


图 2.4 LUSTRE 和 SCADE 中的因果关系

2.5 现状

最后，通过简单回顾 LUSTRE 和 SCADE 的目前研究与发展，可得出以下结论。

首先，仍在进行某些编程语言中的重要改变。这些改进的原型要归功于 Marc Pouzet 的“Lucid synchrone”[11,12]，这是 LUSTRE 的一种更高层次的扩展（或一种 LUSTRE 和 ML 的合并）。

- *数组*。LUSTRE-V4[36] 中介绍的数组机制旨在描述常规电路。这不便于软件实现，因为 V4 数组在目标代码中不能编译为数组（循环更新），目前将这些数组扩展为与数组元素一样多的变量。在文献 [31] 中，提出一种类似于函数编程的具有少数迭代器的全新数组机制。实验表明这种新机制对于大多数实际应用功能足够强大，而且能明显减小代码规模。此外，这种机制还允许采用一种用于包含数组程序通用验证的原始技术[28]。
- *状态机*。早就已知在一些情况下不便于采用数据流方式，如系统描述为自然序列或类似自动机。现也已进行了一些尝试来实现命令和数据流的混合描述[26,29]。目前，在 SCADE 中引入了一种称为“安全状态机”（SSM）的 SYN-

㊀ 根据某些 SCADE 编译操作，用于独立编译和代码可追溯性。其他操作允许与 LUSTRE 中相同的准则。

CCHARTS 简化版本。

- *封装和泛型*。LUSTRE 目前还很少关注封装和泛型等经典研究内容，而这些对于大型程序设计、库定义和复用性是必不可少的。目前，正在引入通用软件包的定义机制；封装常数、数据类型、函数和节点的定义；可能看作静态常量参数、类型函数和节点。终有一天会产生一种“面向对象的 LUSTRE”。

面向分布式结构和事件触发任务（参见 2.3.2.5 节）的 LUSTRE 编译新方法仍利用工业工具来做转换。关于代码生成的另一个课题是内部即时调度；在一些应用中，相对于 I/O 的延时要求，基本循环周期设置过长。在此情况下，想通过改变编译器内部的静态调度，以使得输入采集值能够“接近”计算输出值和排放量。在 ESTEREL 的 SAXO 编译器[38]中可实现这种功能。而对于 LUSTRE，仍在研究[7]。

最后，还利用 LUSTRE 来建模、仿真和验证非同步系统：例如，通过附加输入变量（oracles）引入可控不确定性可实现对所谓的全局异步，局部同步系统（GALS）建模[17]，这可能会受到断言的限制。另外，在 SystemC 的交互层次上进行了关于电路仿真描述的一些实验[32]。

参考文献

1. C. André. Representation and analysis of reactive behaviors: A synchronous approach. In *IEEE-SMC'96, Computational Engineering in Systems Applications*, Lille, France, July 1996.
2. E. A. Ashcroft and W. W. Wadge. LUCID*, The Data-Flow Programming Language*. Academic Press, San Diego, CA, 1985.
3. A. D. Baker, T. L. Johnson, D. I. Kerpelman, and H. A. Sutherland. Grafcet and SFC as factory automation standards. In *American Control Conference*, pp. 1725–1730, 1987.
4. G. Berry and G. Gonthier. The Esterel synchronous programming language: Design, semantics, implementation. *Science of Computer Programming*, 19(2):87–152, 1992.
5. Y. Bertot and P. Castéran. *Interactive Theorem Proving and Program Development—Coq'Art: The Calculus of Inductive Constructions*. Texts in Theoretical Computer Science. An EATCS Series, Springer Verlag, Berlin, 2004.
6. A. Bouajjani, J.-C. Fernandez, and N. Halbwachs. Minimal model generation. In R. Kurshan, ed., *International Workshop on Computer Aided Verification*, Rutgers, New Brunswick, NJ, 1990.
7. P. Caspi, A. Curic, A. Maignan, C. Sofronis, S. Tripakis, and P. Niebert. From Simulink to Scade/Lustre to TTA: A layered approach for distributed embedded applications. In *LCTES 2003*, San Diego, CA, June 2003.
8. P. Caspi, A. Girault, and D. Pilaud. Automatic distribution of reactive systems for asynchronous networks of processors. *IEEE Transactions on Software Engineering*, 25(3):416–427, 1999. Research report INRIA 3491.

9. P. Caspi and N. Halbwachs. A functional model for describing and reasoning about time behaviour of computing systems. *Acta Informatica*, 22:595–697, 1986.
10. P. Caspi, C. Mazuet, R. Salem, and D. Weber. Formal design of distributed control systems with Lustre. In *Proceedings of Safecomp'99, Volume 1698 of Lecture Notes in Computer Science*, Springer Verlag, September 1999.
11. P. Caspi and M. Pouzet. A functional extension to Lustre. In *Eighth International Symposium on Languages for Intensional Programming, ISLIP'95*, Sidney, May 1995.
12. P. Caspi and M. Pouzet. A co-iterative characterization of synchronous stream functions. In *Coalgebraic Methods in Computer Science (CMCS'98)*, Electronic Notes in Theoretical Computer Science, March 28–29, 1998.
13. R. David and H. Alla. *Petri Nets and Grafcet: Tools for Modelling Discrete Event Systems*. Prentice Hall, New York, 1992.
14. L. du Bousquet, F. Ouabdesselam, J.-L. Richier, and N. Zuanon. Lutess: Testing environment for synchronous software. In B.W. Boehm, D. Garlan, and J. Kramer, ed., *Proceedings of the 1999 International Conference on Software Engineering, ICSE '99*. Los Angeles, CA, May, 1999.
15. D. Ferbeck. The VAL product line. In *APM'91 Conference*, Yokohama, 1991.
16. N. Halbwachs. A synchronous language at work: The story of Lustre. In *Third ACM/IEEE International Conference on Formal Methods and Models for Codesign, MEMOCODE'2005*, Verona, Italy, July 2005.
17. N. Halbwachs and S. Baghdadi. Synchronous modeling of asynchronous systems. In *EMSOFT'02*, LNCS 2491, Springer Verlag, October 2002.
18. N. Halbwachs, P. Caspi, P. Raymond, and D. Pilaud. The synchronous dataflow programming language LUSTRE. *Proceedings of the IEEE*, 79(9):1305–1320, 1991.
19. N. Halbwachs, F. Lagnier, and C. Ratel. Programming and verifying real-time systems by means of the synchronous data-flow programming language LUSTRE. In *IEEE Transactions on Software Engineering, Special Issue on the Specification and Analysis of Real-Time Systems*, pp. 785–793, September 1992.
20. N. Halbwachs, F. Lagnier, and P. Raymond. Synchronous observers and the verification of reactive systems. In M. Nivat, C. Rattray, T. Rus, and G. Scollo, eds., *Third International Conference on Algebraic Methodology and Software Technology, AMAST'93*, Workshops in Computing, Springer Verlag, June 20, 1993.
21. N. Halbwachs, Y. E. Proy, and P. Roumanoff. Verification of real-time systems using linear relation analysis. *Formal Methods in System Design*, 11(2):157–185, 1997.
22. N. Halbwachs, P. Raymond, and C. Ratel. Generating efficient code from data-flow programs. In *Third International Symposium on Programming Language Implementation and Logic Programming*, LNCS 528, Springer Verlag, Passau, Germany, August 1991.
23. D. Harel. Statecharts: A visual approach to complex systems. *Science of Computer Programming*, 8(3):231–274, 1987.
24. E. Jahier, P. Raymond, and P. Baufreton. Case studies with Lurette V2. In *First International Symposium on Leveraging Applications of Formal Method, ISoLa 2004*, Paphos, Cyprus, October 2004.
25. B. Jeannet, N. Halbwachs, and P. Raymond. Dynamic partitioning in analyses of numerical properties. In A. Cortesi and G. Filé, eds., *Static Analysis Symposium, SAS'99*, LNCS 1694, Springer Verlag, Venice, Italy, September 1999.

26. M. Jourdan, F. Lagnier, P. Raymond, and F. Maraninchi. A multiparadigm language for reactive systems. In *5th IEEE International Conference on Computer Languages*, IEEE Computer Society Press, Toulouse, May 1994.
27. P. Le Guernic, T. Gautier, M. Le Borgne, and C. Le Maire. Programming real time applications with SIGNAL. *Proceedings of the IEEE*, 79(9):1321–1336, 1991.
28. F. Maraninchi and L. Morel. Arrays and contracts for the specification and analysis of regular systems. In *Fourth International Conference on Application of Concurrency to System Design (ACSD)*, Hamilton, Ontario, Canada, June 2004.
29. F. Maraninchi and Y. Rémond. Mode-automata: About modes and states for reactive systems. In *European Symposium on Programming*, Springer Verlag, Lisbon, Portugal, March 1998.
30. R. Milner. Calculi for synchrony and asynchrony. *Theoretical Computer Science*, 25(3):267–310, 1983.
31. L. Morel. Efficient compilation of array iterators for Lustre. In *First Workshop on Synchronous Languages, Applications, and Programming, SLAP02*, Grenoble, April 2002.
32. M. Moy, F. Maraninchi, and L. Maillet-Contoz. LusSy: A toolbox for the analysis of systems-on-a-chip at the transactional level. In *Fifth International Conference on Application of Concurrency to System Design (ACSD)*, 2005.
33. F. Ouabdesselam and I. Parissis. Testing synchronous critical software. In *5th International Symposium on Software Reliability Engineering (ISSRE'94)*, Monterey, November 1994.
34. J. A. Plaice and J.-B. Saint. The LUSTRE-ESTEREL portable format. Unpublished report, INRIA, Sophia Antipolis, 1987.
35. P. Raymond, D. Weber, X. Nicollin, and N. Halbwachs. Automatic testing of reactive systems. In *19th IEEE Real-Time Systems Symposium*, Madrid, Spain, December 1998.
36. F. Rocheteau and N. Halbwachs. Implementing reactive programs on circuits, a hardware implementation of Lustre. In *REX Workshop on Real-Time: Theory in Practice, DePlasmolen (Netherlands)*, pp. 195–208. LNCS 600, Springer Verlag, June 1991.
37. N. Scaife and P. Caspi. Integrating model-based design and preemptive scheduling in mixed time- and event-triggered systems. In *Euromicro Conference on Real-Time Systems (ECRTS'04)*, Catania, Italy, June 2004.
38. D. Weil, V. Bertin, E. Closse, M. Poisse, P. Venier, and J. Pulou. Efficient compilation of Esterel for real-time embedded systems. In *International Conference on Compilers, Architecture, and Synthesis for Embedded Systems (CASES)*, San Jose, 2000.

第3章 群智能方法形式化集成要求

Mike Hinchey
Lero—爱尔兰软件工程研究中心，利默里克大学，利默里克，爱尔兰
James L. Rash
美国国家航空航天局哥达德太空飞行中心（名誉教授），Greenbelt，马里兰州
Christopher A. Rouff
Near Infinity 公司，雷斯顿，弗吉尼亚州
Walt F. Truszkowski
美国国家航空航天局哥达德太空飞行中心（名誉教授），Greenbelt，马里兰州
Amy K. C. S. Vanderbilt
范德比尔特咨询公司，费尔法克斯，弗吉尼亚州

3.1 简介

美国国家航空航天局（NASA）正在为未来的太空探索研究新的范式，主要集中在自治和自主系统这些新兴技术上[46-48]。目前，正在研究依赖于类似于集群性质的相互协作的多个小型航天器的任务，用于对依靠一个大型航天器的传统任务进行补充和完善[16]。该任务中的各个小型航天器都能够通过自身完成一部分任务，但要成功执行整个任务，则需与其他航天器相互合作并交换各自的信息。

这种新的系统范式具有以下多个优势：

- 具有探索传统飞船无法达到的空间环境和区域的能力；
- 任务冗余度更高（因此，能在更大程度上保护资产）；
- 可降低成本和风险等，在此不一一列举。群体任务的示例中包括：

○ 自主无人机（UAV）在距离火星表面大约1m处飞行，数秒内观测的火星表面面积将等同于目前著名火星探测器所有时间内探索的面积，

○ 利用四面体行走器的手臂去探索火星和月球的表面[15]，

○ 卫星编队飞行星座体系，

○ 小型化微型类航空飞船探索小行星带，迄今为止，不可能在极度高损耗情况下发送探测飞船[16]。

然而，这些新探索方法同时也存在许多挑战。这些任务都是无人操作且高度

自治的。其中，许多都将被发送到太阳系，这对于载人任务，在可预见的技术限制条件下是根本不可能完成的。从地面到航天飞船之间的往返通信延迟将超过40min，这意味着，在探索任务的响应决策以及存在某些问题和无法预料情况下必须在*现场*作出决策而不是从地球上获得地面控制。

这些任务所需的自主性和智能化程度都需要对任何开发系统（软件和硬件），进行大量测试。除此之外，无法完全预测具有学习性和自主性（如自我优化、自我修复）的突发性行为模式，因此，这些任务将需要比传统单个航天器任务复杂程度高多个数量级的命令，并验证这些新类型任务无法利用传统技术实现。笔者认为形式化规范技术和形式化验证技术将在 NASA 太空探索任务的未来发展中发挥重要作用。即使这些行为（一定范围内）是突发的（由于大量相互作用实体的构成，缺少特殊设计和验证措施的产生行为是不可预见的），形式化方法也能够确保软件质量并证明群体行为的正确性。所推导的形式化模型还可作为任务大部分代码自动生成的基础[25]。

为应对这些任务验证过程中面临的挑战，NASA 的一个群体技术形式化方法（FAST）项目研究了用于这些任务的形式化方法[33-37,40,41]。另一个 NASA 的概念任务——自治群体纳米技术（ANTS），作为一个指定和验证任务的示例[15-17]。用于 ANTS 任务中的一种有效形式化方法能够预测 1000 个智能体构成的群体突发行为，也能预测单个智能体的行为。任务完成的关键在于自治性和自主操作调整能力，以反映不断变化的条件和任务目标。

本章首先概述了群体技术，然后基于群体的概念任务 ANTS 给出了用于验证基于群体的任务的大量形式化方法的评估结果，并提出了一种用于验证基于群体的系统的集成形式化方法。

3.2　群体技术

群体是由大量相互作用并与环境交互的简单智能体组成的。其中，没有集中控制器来控制整个群体且没有一个智能体具有全局视图，通过简单的交互作用产生“应急行为”，并动态自组织群体行为。应急行为是由研究社会性昆虫自组织行为的 Bonabeau 等人[8]通过观察昆虫和鸟类而发现的，指出“复杂的集体行为可能是通过具有简单行为的个体之间交互产生的”，并将应急行为描述为“一组通过低级组件之间相互作用而产生系统全局结构的动力学机制”。应急行为有时也称为宏观行为，而个体行为和局部相互作用称为微观行为。

群智能体经常作为一种建模技术和研究复杂系统的一种工具[22]。在群智能体模拟中，主要研究与应急行为相关的一组相互作用的智能体[50]（通常是同构或近似同构）。群智能体模拟在鸟群[11,32]、商业和经济[28]以及生态系统[42]的研

究中得到广泛应用。在群智能体模拟中，给定每个智能体试图使其达到最大化的特定参数。关于鸟群，每只鸟都试图找到另一只鸟来一起飞，并飞到其一边且略微高于以减少阻力，最终形成鸟群。同时，群体研究也用于如电话交换机、网络路由、数据分类和最短路径优化的应用中。

在智能群体中，群个体成员具有智能性[6,7]。智能群中的个体可以是同构或异构。由于各自环境不同，即使初始时是同构的群个体也会学习不同技能并产生不同目标，由此整个群变为异构。开始时为同构的一个群（如下所述的NASA概念任务）将具有不同能力以及可能存在社会结构。由于群体不再是由具有有限智能和通信能力的同构个体组成，这样将使得更加难以验证这些系统。

群体的应急行为可能是不可预知的。尽管群体行为通常是简单个体行为的集合，但是当整合在一起时，可能形成往往意想不到的复杂行为。这使得智能群体验证更加困难，不仅因为每个成员本身更加复杂，而且大量智能个体之间的相互作用也更为复杂。智能群体具有一个巨大的状态空间，并且由于个体可处于“学习模式”，由此，个体行为和群体应急行为可能不断变化而导致难以预测。然而，精确预测这些行为对任务开发人员至关重要，以确保任务操作安全并按预期完成。

本节其余内容是对基于群智能体的概念任务NASA ANTS及其规范难点进行概述。为评估基于群智能体任务中规范、验证和校验的多种形式化方法，在此利用ANTS任务作为实验测试平台和案例分析。

3.2.1 ANTS任务概述

ANTS概念任务[15-17]涉及一群自主微型（约1kg）航天器的发射，通过探索小行星带来寻找具有一定科学性的小行星。图3.1给出了ANTS任务概况[46]。在该任务中，从地球上发射的太空运输飞船将飞行到太空中小型物体（如微型航天器）上的净引力可忽略不计的位置（太阳系中的拉格朗日点）。在该位置处，来自地球上制造的1000个航天器根据不同轨迹发射到小行星带。由于尺寸很小，每个航天器只能携带一个专用仪器来采集行星带中小行星的特定类型数据。

为实现这一任务，考虑采用一种启发式方法来提供具有类似于昆虫聚集地或群体层次化行为的群体社会结构，并由某一航天器指挥其他航天器。研究遗传算法、神经网络、模糊逻辑和车载规划器等人工智能技术来辅助任务保持高度自主性。任务的关键在于应具有能够自主调节操作以体现任务、距离以及与地球通信低带宽不断变化的本质的能力。接近80%的航天器将成为携带专业仪器的“工人”（如磁力计、X射线传感器、伽马射线、可视/红外波段或中性粒子质谱仪），并通过这些仪器采集特定类型的数据。其中某些航天器作为“协调者”

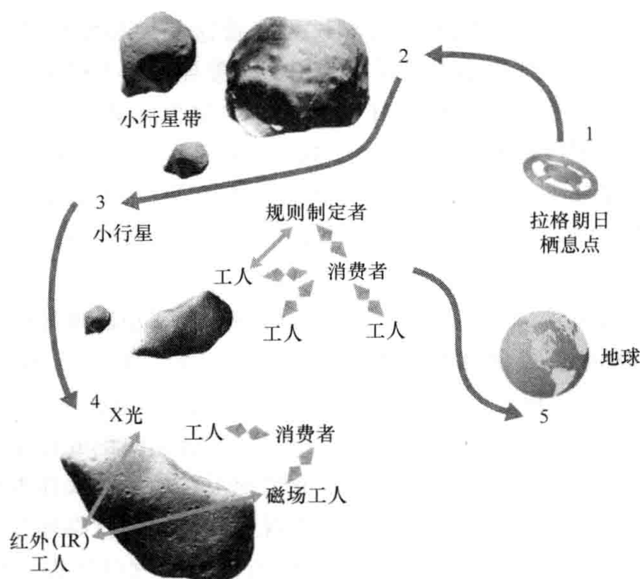


图 3.1　ANTS 任务概念图

（统治者或领导者），具有确定小行星类型和任务感兴趣数据以及协调载体之间工作的规则。第 3 类航天器是“信使”，负责协调统治者和工人之间的通信以及与地球上的任务控制中心（也称为地面控制）进行通信。

当 ANTS 团队遇到小行星时，会发生很多事情。航天器可以飞掠观察并寻找机会进行观测。通过飞掠观察可首先在发送整个团队之前确定该行星是否是感兴趣的行星，并根据航天器搭载的仪器特性，确定是否仅需要飞掠观察。如果确实是目标小行星，则发送一个成像航天器到小行星以确认其确切尺寸和特性，并创建一个用于操控其他航天器探测小行星周围的大致模型。然后，其他航天器团队相互协调以完成对小行星的研究和地图绘制，最终形成一个完整模型。

3.2.2　ANTS 规范和验证

上述只是对 ANTS 任务的一个非常简单的概述。更详细的描述，感兴趣读者可直接参考文献［39］和文献［46］，或浏览 ANTS 网站。从上述的简要概述中可知，ANTS 是一个极具挑战性的高度复杂的系统。其中不仅仅是异构组件之间的复杂相互作用；需要连续的重新规划、重新配置以及重新优化；需要在没有地球控制人员的干预下自主操作；以及需要确保任务的正确操作。

在诸如 ANTS 的高度自治化并与地面控制长时间失联的任务中，一旦启动后，软件错误将是不可观测及或不可纠正的。因此，在任务启动之前，高度确保任务正确是非常有必要的。由于在最终的实际环境下的测试是不切实际或不可能

的，因此需要模拟实现空间探索系统测试。尽管这些模拟实验具有非常高的质量，但通常还是会有一些很小的错误，并由此可能导致整个任务失败。正如在火星极地着陆号任务[5]中所发生的，在飞行软件中缺少可能会导致整个任务失败的一行代码。在火星极地着陆器（Mars Polar Lander）的失败报告中提到：

意识到太空任务是一种“一招不慎，满盘皆输”的行为非常重要。即使正确执行了上万个函数，但只要有一个错误就可能导致任务产生灾难性后果。

因此，

对于任务的成功执行，进行全面彻底的测试和程序验证是必不可少的。

如 ANTS 等复杂任务更加难以发现错误，并需要新的任务验证方法来提供 NASA 所需的软件保证程度，以将风险降低到可以接受的水平。ANTS 的许多行为，包括基于时间产生竞争条件的行为，都只是在特定时刻或特定顺序下执行数据发送和接收才发生，或学习后发生。在此情况下，通过输入样本数据和检查结果正确性几乎无法发现错误。为通过测试发现这些错误，就必须在涵盖软件流程中所有可能的状态组合（状态空间）中执行所涉及的软件处理过程。由于状态空间与状态个数呈指数关系（有时为阶乘关系），即使是相对较少的处理过程，也导致无法测试。传统上，为解决状态爆炸问题，测试人员将人为减少系统中的状态个数，并利用模型来近似所采用的软件程序。这样就减少了模型保真度，并掩盖了潜在错误。

规定（和验证）群体行为的一个重要问题是可支持应急行为的分析与确定。应急思想源自生物学、经济学和其他科学领域，并广为人知，即使在计算机科学与工程领域也是非常有名的。尽管计算机科学家和工程人员经常会遇到应急行为问题，但对于该问题尚未充分理解。应急行为描述为“比个体组件行为更复杂的系统行为，…，且并非按设计者预期执行[49]”。这意味着在充分理解个体行为的系统中，其组件的相互作用与单个环境相结合，可展示出从个体组件行为无法预见或无法解释的行为。由此可推论出，改变系统集中某一系统的组件或替换系统集中某一系统的子系统，都可能对整个系统行为以及其他子系统行为产生不可预见的、意外的、完全无法解释的后果。

群体任务的形式化规范应能够跟踪任务目标的变化，并在获得新数据时可调整通用模型。形式化规范还应允许指定决策过程，以帮助决策需要哪些仪器、放置在何种位置以及具有何种目标等。一旦完成，形式化规范就必须可以证明系统性能，检查特定类型的错误，并作为输入用于模型校验器。形式化方法还必须能够跟踪领导者模型，并当采集数据满足目标要求时，能够作出决策。为实现上述所有功能，将 4 种明确的形式化方法[34,36,37,40,41]集成似乎是验证基于群体系统的最佳方法。

3.3 NASA FAST 项目

FAST 项目确定了验证基于群体系统的形式化方法中所需的几个重要属性[35,37]，并对形式化方法和形式化技术进行广泛调研[34,40,41]以确定现有形式化方法或形式化方法组合是否适用于规定和验证基于群体的任务及其应急行为。根据确定在智能群体应用中非常重要的少数几个标准，对各种形式化方法进行了调研。这些标准包括：

- 支持并发和实时约束；
- 形式化基础；
- （现有的）工具支持；
- 应用于基于智能体或基于群体的系统的以往经验；
- 算法支持。

在此，确定已有大量满足并发行为和算法行为某个规范而并非全部规范的形式化方法。另外，近年来，已开发出大量同时支持并发和算法规范目标的集成形式化方法。根据调研结果，选择 4 种形式化方法用于 ANTS 部分任务的规范示例，这些方法如下：

- 1）通信顺序进程（CSP）[24,26,27]；
- 2）通信系统的加权同步演算（WSCCS）[45]；
- 3）X 机[29-31]；
- 4）Unity（单一）逻辑[13]。

由于本团队在利用 CSP 规范智能体系统方面已具有丰富经验和显著成就，因此在此选择 CSP 作为基准的规范方法。选择 WSCCS 和 X 机是因为已由其他人员成功应用于规范应急行为。选择 Unity 逻辑是因为其已成功应用于规范并发系统，另外，与其他方法相比，该方法是基于逻辑的。将 X 机的记忆功能和迁移功能，与 WSCCS 和其他方法的优先级和概率相集成可产生一种允许规范 ANTS 任务和其他群体系统中应急行为所有必要因素的规范方法。此外，目前已有工具能够支持这些形式化方法[37]。

采用集成形式化方法的做法是一种视点整合方法[18,44]。这种类型的形式化集成方法，保持各方法的形式化基础不变，并研究形式之间的关系来体现新的形式化方法。该方法可保留各个方法的作用，使得实现 ANTS 任务的无缝规范，并允许利用各方法的现有语义来开发相应的支持工具。

视点规范也广泛作为处理规范复杂大系统的一种有效方法。每个视图规范都利用最适合该视图的一种语言来描述系统的某个方面（而不是一个组件或模块）。采用该方法的结果是以不同视图表示相同或相关组件的规范，且必须充分

地相互交叉引用。

3.4　群体形式化集成方法

目前采用的大多数形式化符号都是创建于20世纪70年代和80年代，用于反映当时开发的分布式系统类型。现在，分布式系统不断发展，已不能用规范过去系统的相同方式来进行规范[14]。因此，许多研究人员和从业人员将形式化方法进行组合以形成集成（混合）方法来解决分布式系统的新功能（如移动智能体、群智能体和应急行为）。

集成方法在规范并发系统和基于智能体的系统中得到广泛应用。集成方法通常是在基于模型的方法中结合进程代数或基于逻辑的方法。进程代数或基于逻辑的方法易于规范并发系统，而基于模型的方法则重点在于规范系统中的算法部分。下面列出用于规范并发系统和基于智能体的系统的部分集成方法：

- X机通信[2]；
- CSP-OZ（CSP和Object-Z结合[19]）；
- Object-Z和状态图[10]；
- 定时通信Object-Z[20]；
- 时态B[9]；
- 定时CSP[43]；
- 时序Petri网（时态逻辑和Petri网）[1]；
- ZSSC（Z和CCS结合[21]）。

为阐述CSP、WSCCS、X机和Unity逻辑的集成，下面给出几个不同视图下ANTS任务的示例。在不同视图下表明这些方法如何相互交叉引用以及每个方法如何提供ANTS任务的不同视图。CSP提供进程间通信视图，X机提供状态机和内存视图，WSCCS提供概率和优先级视图，而Unity逻辑提供逻辑视图。其中所引用的每个变量，如目标和模型，都具有额外符号或重点表示其他视图规范的交叉引用。对于较长的规范，可以采用颜色编码。由于这只是用于说明目的的示例规范，因此采用粗体突出显示。以下各小节给出了基于文献［36，37，41］中所列规范的视图示例。

3.4.1　CSP简述

以下是CSP方法中ANTS任务的顶层规范：

$$ANTS_{\text{goals}} = Leader_{i,\text{l_goals}} \| Messenger_{j,\text{m_goals}} \| Worker_{k,\text{w_goals}}$$

$$1 \leqslant i \leqslant m,\ 1 \leqslant j \leqslant n,\ 1 \leqslant k \leqslant p$$

式中，m 为领导者航天器个数；n 为信使航天器个数；p 为工人航天器个数。

该规范表明根据由主要研究人员给定的一组目标开始或初始化 ANTS 任务，其中一些目标是针对领导者（而一些目标并非针对领导者，是因为这些目标是地面控制或不适合于领导者的）。除了目标，每个航天器还会给定一个名称（在本例下，通常是数字形式），以便可以在与其他 ANTS 航天器和地球通信时自身识别。

对于视点整合，ANTS 的目标也将由 ANTS 的 X 机视图指定。在此突出显示目标以表示在其他视图中也被引用。下面给出领导者的通信规范并表明在其他视图中使用的规范和变量。

3.4.1.1　领导者规范

领导者航天器规范由通信进程和智能进程组成：

$$Leader_i = LEADER_COM_i,\ \{\}\ \|LEADER_INTELLIGENCE_{i,\mathrm{goals},\mathrm{model}}$$

通信进程，*LEADER_COM*，用于规范在与其他航天器和地球进行通信时的航天器行为，并规范航天器之间的协议。第 2 个进程，*LEADER_INTELLIGENCE*，表示领导者智能，即保证航天器及其环境的目标和模型，并规范运行过程中如何修改目标和模型。对于上述的每个进程，一个重要参数是航天器组标识符，而其他参数分别设置存储会话（开始时为空）、目标和模型。在 X 机规范中指定这些目标和模型，而 *LEADER_INTELLIGENCE* 进程的推理部分在 WSCCS 规范中指定。

下面给出领导者通信进程中的规范。

3.4.1.2　领导者通信规范

一个领导者航天器可接收来自另一个领导者、信使、工人或地球的消息。若来自其他发送者的消息中包含错误信息，则返回该消息（假设具有一种保证错误消息返回的机制，从而导致无限循环）。假设从另一个航天器中继的消息标记为来自于最初发送者，而非信使。领导者还会维持一组持久消息来追踪当前的会话状态。领导者可请求获得其他航天器状态，移动到新位置，更改目标，或返回特定数据。另外，还应发送给其他航天器类似行为的请求信息。

通过 WSCCS、X 机和 Unity 逻辑来规范领导者通信，并在进行通信时，根据推理状态或进程状态来改变状态。另外，如果通信时，在 WSCCS、X 机或 Unity 逻辑规范中发生任一所列动作，则会发生从通信状态到推理状态或进程状态的变化。再次利用突出显示来表明在其他视图中的规范。以下每个 CSP 规范的相交还可作为迁移到 WSCCS、X 机或 Unity 逻辑有限状态机的一种行为。

以下是领导者通信的顶层规范：

$LEADER_COM_{i,conv} = leader.\ in?\ msg \rightarrow$

case $\mathrm{LEADER_MESSAGE}_{i,conv,msg}$

if *sender* (*msg*) = *LEADER*

$\text{MESSENGER_MESSAGE}_{i,conv,msg}$

if *sender* (*msg*) = *MESSENGER*

$\text{WORKER_MESSAGE}_{i,conv,msg}$

if *sender* (*msg*) = *WORKER*

$\text{EARTH_MESSAGE}_{i,conv,msg}$

if *sender* (*msg*) = *EARTH*

$\text{ERROR_MESSAGE}_{i,conv,msg}$

otherwise

上述表明一个领导者可能接收到的来自其他航天器的消息类型。WSCCS、X机和 Unity 逻辑视图将上述“LEADER_MESSAGE”、“MESSENGER_MESSAGE”、“WORKER_MESSAGE”、“EARTH_MESSAGE”和“ERROR_MESSAGE”定义为从推理状态或进程状态转换为通信状态的行为。Unity 逻辑的规范中也有表示传递消息条件的谓词。突出显示上述描述以突出显示与另一个视图的链接。

下面进程进一步描述了其他领导者所接收的消息。由其他领导者发送的消息可能为两种类型之一：请求或信息。可能会为获得领导者模型或目标信息、资源信息或状态信息而发布请求。消息也是一种信息，其中包含有关智能体模型新目标或新信息的数据（如一个新发现或来自地球的消息）。该信息需要由智能进程和模型进程来检验，以确定是否需要更新目标或模型。由于 X 机也能规范目标和模型，因此每个引用都链接到 X 机规范：

$\text{LEADER_MESSAGE}_{i,conv}$ =

case $LEADER_INFORMATION_{i,conv,msg}$

if *content* (*msg*) = *information*

$LEADER_REQUESTS_{i,conv,msg}$

if *content* (*msg*) = *request*

$LEADER_RECEIVE_{i,conv,msg}$

if *content* (*msg*) = *reply_to_request*

$ERROR_MESSAGE_{i,conv,msg}$

otherwise

上述每个子进程的进一步规范如下：

$LEADER_INFORMATION_{i,conv}$ =

leader_model_i! (*NEW_INFO*, *msg*)

$\rightarrow \text{goals_channel}_i$! (*NEW_INFO*, *msg*)

$\rightarrow \mathrm{LEADER_COM}_{i,conv}$

如果消息为新信息，那么就应将信息发送到智能体的代理部分以检查是否应该更新目标，以及发送到智能体的模型部分以检查是否需要更新模型。再次强调，由于在其他视图（X 机）中已定义模型和目标以及通信造成在 WSCCS、X 机和 Unity 逻辑规范中状态改变，因此突出显示上述过程：

$$
\begin{aligned}
& LEADER_REQUESTS_{i,conv,msg} = \\
& \quad \text{case } LEADER_STATUS_REQ \\
& \qquad \text{if } content\ (msg) = status_request \\
& \quad LEADER_INFO_REQ_{i,conv,msg} \\
& \qquad \text{if } content\ (msg) = info_request \\
& \quad LEADER_RESOURCE_REQ_{i,conv,msg} \\
& \qquad \text{if } content\ (msg) = resource_request \\
& \quad ERROR_MESSAGE_{i,conv,msg} \\
& \qquad \text{otherwise}
\end{aligned}
$$

如果消息为一个请求，则根据请求类型，执行不同的进程。一个智能体（如工人）可能会请求获得航天器状态信息、领导者目标或模型信息，或资源信息（如在领导者指挥下工人应形成一个小组来调查一个特定小行星，或应重新定位一个信使以完成通信功能）。由于该规范是底层规范的抽象，因此并不影响其他视图中的规范：

$$
\begin{aligned}
& LEADER_STATUS_REQ_{i,conv,msg} = \\
& \quad \mathrm{leader}_i!\ \mathrm{reply}\ (msg,\ current_status\ (\,)) \\
& \rightarrow LEADER_COM_{i,conv}
\end{aligned}
$$

如上所述，如果为状态请求，则通过利用一个状态检索的标准函数，将航天器当前状态返回请求方。规范中的 $leader_i!\ reply$ 是一种导致 WSCCS、X 机和 Unity 逻辑规范中状态改变的通信：

$$
\begin{aligned}
& LEADER_INFO_REQ_{i,conv,msg} = \\
& \quad \text{case goals_channel}_i!\ (REQUEST,\ msg) \rightarrow LEADER_COM_{i,conv} \\
& \qquad \text{if } content\ (msg) = goals_request \\
& \quad \text{leader_model}_i!\ (REQUEST,\ msg) \rightarrow LEADER_COM_{i,conv} \\
& \qquad \text{if } content\ (msg) = model_request \\
& \quad ERROR_MESSAGE_{i,conv} \\
& \qquad \text{otherwise}
\end{aligned}
$$

对于 *LEADER_INFO_REQ* 进程，如果为领导者目标信息请求，则将该消息

发送到领导者智能进程以检索该信息，然后返回到请求者。如果为领导者模型信息请求，则将该请求发送到领导者模型进程，然后将模型信息返回到请求者。目标和模型作为X机规范中的一部分，且通信会造成在WSCCS、X机和Unity逻辑规范中的状态改变：

$LEADER_RESOURCE_REQ_{i,conv,msg} =$

goals_channel$_i$!（$RESOURCE$，msg）

$\rightarrow LEADER_COM_{i,conv}$

对于资源请求，必须检查领导者的目标以确定放弃该资源是否会影响领导者的当前目标。因此，将该消息发送到智能进程来检查当前目标，更新其目标和模型（在放弃资源的情况下），并适当回复请求者。突出显示的过程表明在X机视图指定目标，且通信会导致WSCCS、X机和Unity逻辑规范中的状态变化：

$LEADER_RECEIVE_{i,conv,msg} =$

case LEADER_STATUS_RECEIVED$_{i,conv,msg}$

if $content$（msg）$= status_returned$

LEADER_INFO_RECEIVED$_{i,conv,msg}$

if $content$（msg）$= info_returned$

LEADER_RESOURCE_RECEIVED$_{i,conv,msg}$

if $content$（msg）$= resource_request_return$

ERROR_MESSAGE$_{i,conv,msg}$

otherwise

在对其他实体发送一个请求后，领导者将接收状态、信息或资源请求的返回消息。上述 *LEADER_RECEIVE* 是处理返回消息的进程。这些消息必须发送到领导者的审议部分，由此才能更新领导者的目标和模型。通信改变了其他视图中的状态机，且在Unity逻辑规范中也定义了从其他航天器接收消息的条件：

$LEADER_STATUS_RECEIVED_{i,conv,msg} =$

goals_channel$_i$!（$STATUS_RECEIVED$，msg）

$\rightarrow LEADER_COM_{i,conv}$

当领导者接收到来自另一领导者的状态消息时，执行 *LEADER_STATUS_RECEIVED* 进程。在此情况下，将消息发送到包含该消息的目标过程，以便可更新目标和模型。如果状态表明工人无法执行部分功能，则需要更新该目标。突出显示该过程以表明X机视图中的规范：

$LEADER_INFO_RECEIVED_{i,conv,msg} =$

goals_channel$_i$!（$INFO_RECEIVED$，msg）

$\rightarrow LEADER_COM_{i,conv}$

当领导者接收到来自另一领导者的信息消息时，执行 *LEADER_INFO_RECEIVED* 进程。同上，将消息发送到目标进程，以检查是否需要更新目标和模型。同时突出显示该过程以表明不同视图（X 机）中的规范：

$$LEADER_RESOURCE_RECEIVED_{i,conv,msg} = \\ \quad \text{goals_channel}_i!\ (RESOURCE_RECEIVED,\ msg) \\ \quad \rightarrow LEADER_COM_{i,conv}$$

根据上述进程，当执行 *LEADER_RESOURCE_RECEIVED* 进程时，消息发送到领导者的目标进程，以确定是否更新目标和模型，如果需要则采取相应动作。突出显示该过程以表明不同视图（X 机）中的规范：

$$ERROR_MESSAGE_{i,conv,msg} = \\ \quad \text{messenger}_i!\ \text{return}\ (msg,\ error) \\ \quad \rightarrow LEADER_COM_{i,conv}$$

如果消息不能与任何其他交互匹配，则向消息发送航天器返回一个错误消息。突出显示该错误消息，因为通信会导致 WSCCS、X 机和 Unity 逻辑视图中状态改变。

从信使发送到领导者的消息和从工人发送到领导者的消息具有与上述类似的规范。参见文献［34，40，41］。

3.4.2 WSCCS 简述

为建立 ANTS 中的领导者航天器模型，采用一种进程代数 WSCCS 时应考虑：

- 领导者的可能状态（智能体）；
- 使得智能体“位于”某些状态的各个智能体状态中的动作；
- 为智能体定义的每个动作的相对频率；
- 为智能体定义的每个动作的优先级。

3.4.2.1 动作

表 3.1 给出了领导者的动作、智能体状态和动作优先级。所有这些动作都体现在其他规范视图中。

由于规范中的状态部分和 X 机、Unity 逻辑规范类似，WSCCS 中状态的任何变化都能影响 X 机规范中的内存和状态变化以及 Unity 逻辑中的状态机，并受到 Unity 逻辑谓词的影响。上述强调的动作会导致按照 CSP 视图中的规范迁移到通信状态，并表明 CSP 规范何时会使得数据通过 CSP 通道。此外，Unity 逻辑的谓词指定其应用的前提条件。推理和处理部分与规范中 *LEADER_INTELLIGENCE* 的 CSP 规范有关（在本书中并未包含，可参见文献［34，40，41］）。

表 3.1 智能体状态和动作

智能体状态	导致智能体状态变化的动作	f	p
通信	识别		
	发送工人消息	50	2
	发送领导者消息	50	2
	发送错误消息	1	1
	接受工人消息	50	2
	接收领导者消息	50	2
	接收错误消息	1	1
推理	推理审议	50	2
	推理响应	50	2
处理	排序和存储处理	17	2
	生成处理	17	2
	预测处理	17	2
	诊断处理	16	2
	恢复处理	16	2
	修复处理	17	2

继续采用 WSCCS 规范，根据表 3.1 中的给定信息，定义智能体不同状态：

$$AgentD \equiv n_a: a.AgentA + n_b: b.AgentB + n_c: c.AgentC$$

式中，n_a是形式 $n\omega^k$的权重，n 为动作 a 的相对频率，k 表示动作 a 的优先级，将使得智能体 D 变为智能体 A。

另外，还表示在可能动作之间的一种选择。由此，智能体 D 可选择执行动作 a，将智能体 D 变为智能体 A。智能体 D 以频率 n 和优先级 k 进行该选择。或智能体 D 可选择执行动作 b 等。利用该符号，领导者具有如下表示的智能体状态：

$$\begin{aligned}
&Communicating \equiv \\
&50\omega^3: Reasoning\ Deliberative.Reasoning \\
&+50\omega^3: Reasoning\ Reactive.Reasoning \\
&+50\omega^2: Reasoning\ Deliberative.Reasoning \\
&+17\omega^3: Processing\ Sorting\ And\ Storage.Processing \\
&+17\omega^3: Processing\ Generation.Processing \\
&+17\omega^3: Processing\ Prediction.Processing \\
&+16\omega^4: Processing\ Diagnosis.Processing
\end{aligned}$$

$+16\omega^4$: *Processing Recovery. Processing*

$+17\omega^4$: *Processing Remediation. Processing*

根据这种表示，当处于通信状态时，领导者可选择（允许）执行以下集合中的任何动作：

{*ReasoningDeliberative*，*ReasoningReactive*，
ProcessingSortingAndStorage，*ProcessingGeneration*，
ProcessingPrediction，*ProcessingDiagnosis*，*ProcessingRecovery*，
ProcessingRemediation}

并且通信领导者将以 25% 的概率（上述所有总频率）执行 *ReasoningDeliberative*，且给定该动作的优先级为 3。上述表示的第 2 项说明通信领导者以相同 25% 的概率执行 *ReasoningReactive* 且优先级也为 3。符号 + 表示通信领导者将在各种允许动作之间进行选择，这种选择是基于每个允许动作的频率和优先级的。WSCCS 是唯一一个考虑状态变化的概率和优先级的视图。概率和优先级可允许计算系统的稳定状态，以及何种应急行为。

对于一个给定智能体，迁移语义定义了有效动作序列，并允许通过一个节点表示智能体，节点之间的连线（边界）表示权重和动作的迁移图来将智能体解释为有限状态自动机。

智能体迁移定义为

$$D\ulcorner\{a, b, c\} \xrightarrow{a\left[\frac{n_a}{n}\right]} A\ulcorner\{a, b, c\}$$

式中，$a[n_a/n]$ 为动作 a 的发生概率；n 为可能动作 a、b、c 相对频率之和。

考虑上述迁移，该迁移定义表示智能体 D 只能执行动作 a、b 或 c。智能体 D 执行动作 a 的概率为 $[n_a/n]$，且动作结果是智能体 D 变为仅执行动作 a、b 或 c 的智能体 A。

以下是关于群体通信部分（Communicating）的规范。其余的通信部分以及处理和推理部分均类似：

Communicating ⌈{*ReasoningDeliberative*，*ReasoningReactive*，
ProcessingSortingAndStorage，*ProcessingGeneration*，
ProcessingPrediction，*ProcessingDiagnosis*，*ProcessingRecovery*，

$$ReasoningReliverative\left[\frac{50\omega^3}{200}\right]$$

ProcessingRemediation} $\longrightarrow$
Reasoning ⌈{*SendMessageWorker*，*SendMessageWorkerVIAMessenger*，
SendMessageLeader，*SendMessageLeaderVIAMessenger*，

SendMessageError，*ReceiveMessageWorker*，
ReceiveMessageWorkerVIAMessenger，*ReceiveMessageLeader*，
ReceiveMessageLeaderVIAMessenger，*ReceiveMessageError*，
ProcessingSortingAndStorage，*ProcessingGeneration*，
ProcessingPrediction，*ProcessingDiagnosis*，*ProcessingRecover*，
ProcessingRemediation}

上述通信领导者的表示规范了对于动作集合

{*ReasoningDeliberative*，*ReasoningReactive*，
ProcessingSortingAndStorage，*ProcessingGeneration*，
ProcessingPrediction，*ProcessingDiagnosis*，*ProcessingRecovery*，
ProcessingRemediation}

通信领导者将以 50% 的概率且优先级为 3 来选择动作 *ReasoningDeliberative*，且在执行动作时，通信领导者将转换为推理领导者，以允许从如下给出的动作集中做出选择，然后迁移到另一状态：

{*SendMessageWorker*，*SendMessageWorkerVIAMessenger*，
SendMessageLeader，*SendMessageLeaderVIAMessenger*，
SendMessageError，*ReceiveMessageWorker*，
ReceiveMessageWorkerVIAMessenger，*ReceiveMessageLeader*，
ReceiveMessageLeaderVIAMessenger，*ReceiveMessageError*，
ProcessingSortingAndStorage，*ProcessingGeneration*，
ProcessingPrediction，*ProcessingDiagnosis*，*ProcessingRecovery*，
ProcessingRemediation}

对于其他视图，每次迁移时，CSP、X 机和 Unity 逻辑规范将表明如何进行通信以及通信条件。

3.4.2.2 迁移图

一个由 ANTS 领导者航天器状态迁移推导而得的迁移图如图 3.2 所示［节点表示智能体，节点之间的连线（边界）代表权重和动作］。

3.4.3 X 机

X 机定义为以下元组：

$$L = (Input,\ Memory,\ Output,\ Q,\ \Phi,\ F,\ start,\ m_0)$$

式中，元组元素定义如下：

- *Input*、*Memory* 和 *Output* 为数据集；

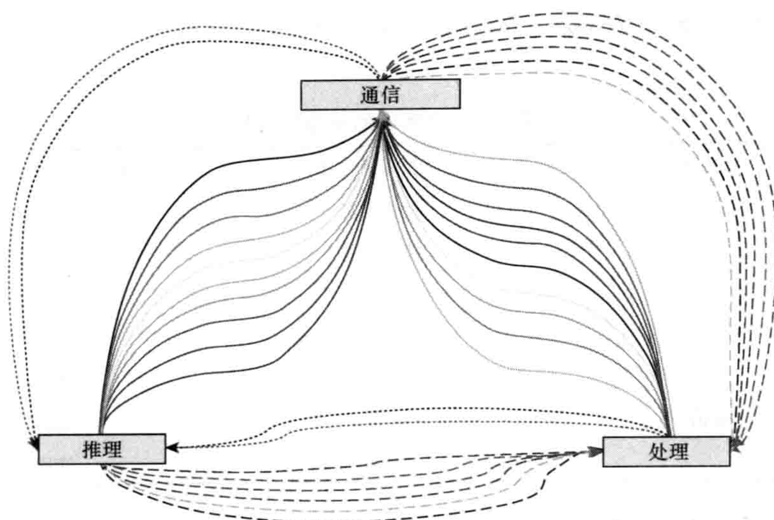


图 3.2　WSCCS 规范中显示权重和动作的迁移图

- Q 为一个有限状态集；
- Φ 是一组（部分）迁移函数，其中每个迁移函数映射为 $Memory \times Input \rightarrow Output \times Memory$；
- F 是下一个状态的部分功能，$F: Q \times \Phi \rightarrow Q$；
- $start \in Q$ 为初始状态；
- $m_0 \in Memory$ 是内存初始值。

对于 ANTS 规范，定义 X 机组件如下：

- *Input* = {*worker*, *messenger*, *leader*, *error*, *Deliberative*, *Reactive*, *SortAndStore*, *Generate*, *Predict*, *Diagnose*, *Recover*, *Remediate*}。
- 内存记为一个元组 $m = (Goals, Model)$，其中 *Goals* 描述任务目标，*Model* 表示领导者保持的通用模型。初始内存由 $(Goals_0, Model_0)$ 表示。当目标和/或模型发生变化时，将新的元组表示为 $m' = (Goals', Model')$。
- *Output* = {*SentMessageWorker*, *SentMessageMessenger*, *SentMessageLeader*, *SentMessageError*, *ReceivedMessageWorker*, *ReceivedMessageMessenger*, *ReceivedMessageLeader*, *ReceivedMessageError*, *ReasonedDeliberatively*, *ReasonedReactively*, *ProcessedSortingAndStoring*, *ProcessedGeneration*, *ProcessedPrediction*, *ProcessedDiagnosis*, *ProcessedRecovery*, *ProcessedRemediation*}。
- Q = {*Start*, *Communicating*, *Reasoning*, *Processing*}。

• Φ = {*SendMessage*，*ReceiveMessage*，*Reason*，*Process*} 函数在表 3.2 中定义。

表 3.2 指挥者的状态和迁移

状态 Q	智能体状态 Φ	$Q'=F\ (Q,\ \Phi)$
开始	发送消息	通信
	接收消息	通信
	推理	推理
	处理	处理
	发送消息	通信
通信	接收消息	通信
	推理	推理

为在这些术语中观察领导者航天器，参照表 3.3，描述状态、迁移函数以及相关的输入、输出和存储单元。ANTS 领导者航天器的一个迁移图如图 3.3 所示[节点表示状态，节点之间的连线（边界）表示迁移函数]。

表 3.3 ANTS 的角色：领导者航天器

状态	迁移函数 $F(Q,\ \Phi)$	函数定义 $\Phi(m,\ \sigma)$ =
开始	发送消息	Φ（m，*Worker*）=（m'，*SentMessageWorker*）
		Φ（m，*Messenger*）=（m'，*SentMessageMessenger*）
		Φ（m，*Leader*）=（m'，*SentMessageLeader*）
		Φ（m，*Error*）=（m'，*SentMessageError*）
通信	接收消息	Φ（m，*Worker*）=（m'，*SentMessageWorker*）
		Φ（m，*Messenger*）=（m'，*SentMessageMessenger*）
		Φ（m，*Leader*）=（m'，*SentMessageLeader*）
		Φ（m，*Error*）=（m'，*SentMessageError*）
推理	推理	Φ（m，*Deliberative*）=（m'，*ReasonedDeliberatively*）
		Φ（m，*Reactive*）=（m'，*ReasonedDeactively*）
处理	处理	Φ（m，*SortAndStore*）=（m'，*ProcessedSortingAndStoring*）
		Φ（m，*Generate*）=（m'，*ProcessedGeneration*）
		Φ（m，*Predict*）=（m'，*ProcessedPrediciton*）
		Φ（m，*Diagnose*）=（m'，*ProcessedDiagnosis*）
		Φ（m，*Recover*）=（m'，*ProcessedRecovery*）
		Φ（m，*Remediate*）=（m'，*ProcessedRemediation*）

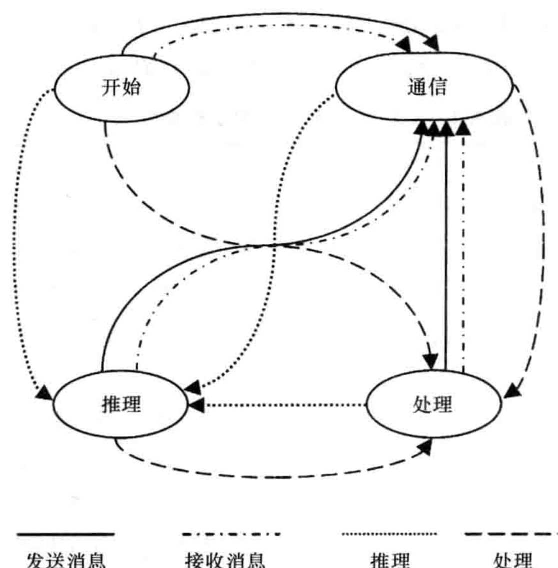


图 3.3　作为 X 机的领导者航天器迁移图

对于视图规范的相互作用，上述所有情况都会影响 CSP 规范、WSCCS 规范，或 Unity 逻辑规范。由于状态相同（除了开始状态），任何时候在 X 机规范中的状态变化、WSCCS 规范中的优先级和概率，以及 Unity 逻辑状态机和谓词都应进行检查。由于之间具有相似之处，该规范可与增加的目标和模型数据相结合。对于 CSP 视图，在通信状态变化的任何时候，CSP 中的通信规范将会作用，尤其是通道上传送的数据。同理，对于 Unity 逻辑，只有发生通信，逻辑谓词将影响迁移是否发生。

3.4.4　Unity 逻辑

在利用 Unity 逻辑来建立 ANTS 的领导者模型时，认为领导者状态与在 WSCCS、X 机以及其他基于状态机的规范语言中的状态相同。在 Unity 逻辑中，将重点考虑领导者的状态以及使得领导者到达这些状态所采取的动作，而所用符号与经典逻辑中非常相似。

谓词定义为表示将领导者置于不同状态的动作（见表 3.4）。然后，这些谓词表示如果为真，意味着领导者已执行使其到达相应状态的动作（见表 3.5）。

可利用下列声明来规范领导者程序的通信部分（推理和处理类似）：

[*Communicating*]　*ReasoningDeliberative*　(*Leader*)[*Reasoning*]

[*Communicating*]　*ReasoningReactive*　(*Leader*)[*Reasoning*]

[*Communicating*]　*ProcessingSortingAndStorage*　(*Leader*)[*Processing*]

[*Communicating*] *ProcessingGeneration* (*Leader*)[*Processing*]
[*Communicating*] *ProcessingPrediction* (*Leader*)[*Processing*]

表 3.4 X 机迁移图中的谓词和含义（一）

谓　语	含　义
SendMessage（Leader，Worker）	领导者向工人发送一条消息
SendMessageVIAMessenger（Leader，Worker）	领导者通过信使中继向工人发送一条消息
SendMessage（Leader，Leader）	领导者向另一个领导者发送一条消息
SendMessageVIAMessenger（Leader，Leader）	领导者通过信使中继向另一领导者发送一条消息
SendMessageError（Leader）	领导者发送一条错误消息
ReceiveMessage（Leader，Worker）	领导者从工人处接收一条消息
ReceiveMessageVIAMessenger（Leader，Worker）	领导者通过信使中继从工人处接收一条消息
ReceiveMessage（Leader，Leader）	领导者从另一领导者处接收一条信息
ReceiveMessageVIAMessenger（Leader，Leader）	领导者通过信使中继从另一领导者处接收一条信息
ReceiveMessageError（Leader）	领导者接收一条错误信息
ReasoningDeliberative（Leader）	领导者慎思式推理
ReasoningReactive（Leader）	领导者反应式推理
ProcessingSortingAndStorage（Leader）	领导者处理数据排序、分类和/或存储
ProcessingGeneration（Leader）	领导者通过模型生成进行处理
ProcessingPrediction（Leader）	领导者通过预测小行星特性或预测资源（工人和通信）可用性进行处理
ProcessingDiagnosis（Leader）	领导者为诊断进行处理
ProcessingRecovery（Leader）	领导者为恢复进行处理
ProcessingRemediation（Leader）	领导者为补救进行处理

表 3.5 X 机迁移图中的谓词和含义（二）

程序状态	如果为真，导致程序状态的语句
	SendMessage（Leader，Worker）
	SendMessageVIAMessenger（Leader，Worker）
	SendMessage（Leader，Leader）
	SendMessageVIAMessenger（Leader，Leader）
	SendMessageError（Leader）
通信	ReceiveMessage（Leader，Worker）
	ReceiveMessageVIAMessenger（Leader，Worker）

（续）

通信	ReceiveMessage（Leader，Leader） ReceiveMessageVIAMessenger（Leader，Leader） ReceiveMessageError（Leader）
推理	ReasoningDeliberative（Leader） ReasoningReactive（Leader） ProcessingSortingAndStorage（Leader） ProcessingGeneration（Leader）
处理	ProcessingPrediction（Leader） ProcessingDiagnosis（Leader） ProcessingRecovery（Leader） ProcessingRemediation（Leader）

[Communicating] ProcessingDiagnosis（Leader）[Processing]
[Communicating] ProcessingRecovery（Leader）[Processing]
[Communicating] ProcessingRemediation（Leader）[Processing]

接下来，Unity 逻辑提供一个等价于命题逻辑的逻辑语法，用于推理其中所隐含的谓词和状态，以及定义执行数学计算、统计计算和其他简单计算的规范。对于视图规范，谓词可定义迁移发生的条件。上述规范的状态机与 WSCCS 和 X 机的状态机类似，可相互结合。

3.5 小结

该项目表明集成 CSP、WSCCS、X 机和 Unity 逻辑的形式化方法如何规范和预测，如 ANTS 任务的基于群体任务的应急行为。由于 CSP 可提供进程间通信视图、X 机可提供状态机和内存视图、WSCCS 可提供概率和优先级视图以及 Unity 逻辑可提供逻辑视图，利用视图集成，这些形式化方法可提供验证航天器群或其他基于群体的系统所需的条件。

这 4 种形式化方法之间具有重叠，特别是 WSCCS、X 机和 Unity 逻辑在状态机方面的重叠。因此，*保护集成或单片集成*可提供一种更加集成的规范。单片集成是首先返回基础形式（通常为语言的一阶逻辑定义），然后与基础形式相融合，并重新定义形式语义。对于保护集成，通过保留基础形式而松散集成[44]。尽管在该方向上的预算和时间不足以完成研究工作，但这种类型的集成可减少视图集成中的重叠，并提供更紧凑的形式化集成方法，如上所述，充分利用各个形式化方法的优势来实现互补。

致谢

本工作得到了 NASA 安全与任务保障办公室（OSMA）的软件保障研究项目（SARP）的资助，并由 NASA 独立验证和检验(IV&V）机构管理实施。另外，本工作的部分内容还得到 Lero—爱尔兰软件工程研究中心的爱尔兰科学基金 03/CE2/I303_1 的资助。需要说明的是，本章内容在很大程度上基于文献［34］。

参考文献

1. I. Bakam, F. Kordon, C. L. Page, and F. Bousquet. Formalization of a spatialized multiagent model using coloured Petri nets for the study of a hunting management system. In *Proceedings of the First International Workshop on Formal Approaches to Agent-Based Systems (FAABS I)*, number 1871 in LNAI, Springer, Greenbelt, MD, April 2000.
2. J. Barnard, J. Whitworth, and M. Woodward. Communicating X-machines. *Information and Software Technology*, 38(6):401–407, 1996.
3. G. Beni. The concept of cellular robotics. In *Proceedings of the 1988 IEEE International Symposium on Intelligent Control*, pp. 57–62. IEEE Computer Society Press, Los Alamitos, CA, 1988.
4. G. Beni and J. Want. Swarm intelligence. In *Proceedings of the Seventh Annual Meeting of the Robotics Society of Japan*, pp. 425–428. RSJ Press, Tokyo, Japan, 1989.
5. M. Blackburn, R. Busser, A. Nauman, R. Knickerbocker, and R. Kasuda. Mars Polar Lander fault identification using model-based testing. In *Proceedings of the 26th Annual IEEE/NASA Software Engineering Workshop (SEW)*, Greenbelt, MD, December 2001.
6. E. Bonabeau, M. Dorigo, and G. Théraulaz. *Swarm Intelligence: From Natural to Artificial Systems*. Oxford University Press, New York, 1999.
7. E. Bonabeau and G. Théraulaz. Swarm smarts. *Scientific American*, 282:72–79, 2000.
8. E. Bonabeau, G. Théraulaz, J.-L. Deneubourg, S. Aron, and S. Camazine. Self-organization in social insects. *Trends in Ecology and Evolution*, 12:188–193, 1997.
9. L. Bonnet, G. Florin, L. Duchien, and L. Seinturier. A method for specifying and proving distributed cooperative algorithms. In *Proceedings of the DIMAS-95*, November 1995.
10. R. Büssow, R. Geisler, and M. Klar. Specifying safety-critical embedded systems with Statecharts and Z: A case study. In Astesiano, ed., *Proceedings of the International Conference on Fundamental Approaches to Software Engineering*, pp. 71–87, number 1382 in LNCS, Springer-Verlag, Berlin, Germany, 1998.
11. S. Carlson. Artificial life: Boids of a feather flock together. *Scientific American*, 283: 112–114, 2000.
12. J. Casani, C. Whetsler, A. Albee, S. Battel, R. Brace, G. Burdick, P. Burr, D. Dippoey, J. Lavell, C. Leising, D. MacPherson, W. Menard, R. Rose, R. Sackheim, and A. Schallenmuller. Report on the loss of the Mars Polar Lander and Deep Space 2

missions. Technical Report JPL D-18709, Jet Propulsion Laboratory, California Institute of Technology, 2000.

13. K. M. Chandy and J. Misra. *Parallel Program Design: A Foundation*. Addison-Wesley Publishing Company, Boston, 1988.
14. E. M. Clare and J. M. Wing. Formal methods: State of the art and future directions. *ACM Computing Surveys*, 28(4):626–643, 1996.
15. P. E. Clark, S. A. Curtis, and M. L. Rilee. ANTS: Applying a new paradigm to Lunar and planetary exploration. In *Proceedings of the Solar System Remote Sensing Symposium*, Pittsburgh, PA, September 20–21, 2002.
16. S. A. Curtis, J. Mica, J. Nuth, G. Marr, M. L. Rilee, and M. K. Bhat. ANTS (Autonomous Nano-Technology Swarm): An artificial intelligence approach to asteroid belt resource exploration. In *Proceedings of the International Astronautical Federation, 51st Congress*, October 2000.
17. S. A. Curtis, W. F. Truszkowski, M. L. Rilee, and P. E. Clark. ANTS for the human exploration and development of space. In *Proceedings of the IEEE Aerospace Conference*, Big Sky, MT, March 9–16, 2003.
18. J. Derrick, E. Boiten, H. Bowman, and M. Steen. Supporting ODP—Translating LOTOS to Z. In *First IFIP International Workshop on Formal Methods for Open Object-Based Distributed Systems*, pp. 399–406. Chapman & Hall, 1996.
19. C. Fischer. Combination and implementation of processes and data: From CSP-OZ to Java. PhD thesis, Universität Oldenburg, Germany, 2000.
20. A. K. Gala and A. D. Baker. Multi-agent communication in JAFMAS. In *Proceedings of the Workshop on Specifying and Implementing Conversation Policies, Third International Conference on Autonomous Agents (Agents '99)*, Seattle, WA, 1999.
21. A. J. Galloway and W. J. Stoddart. An operational semantics for ZCCS. In M. Hinchey and S. Liu, eds., *Proceedings of the IEEE International Conference on Formal Engineering Methods (ICFEM-97)*, pp. 272–282, Hiroshima, Japan, November 1997. IEEE Computer Society Press, Los Alamitos, CA.
22. D. E. Hiebeler. The swarm simulation system and individual-based modeling. In *Proceedings of the Decision Support 2001: Advanced Technology for Natural Resource Management*, Toronto, Canada, September 1994.
23. M. Hinchey, J. Rash, and C. Rouff. Verification and validation of autonomous systems. In *Proceedings of the SEW-26, 26th Annual NASA/IEEE Software Engineering Workshop*, pp. 136–144, Greenbelt, MD, November 2001. NASA Goddard Space Flight Center, Greenbelt, MD, IEEE Computer Society Press, Los Alamitos, CA.
24. M. G. Hinchey and S. A. Jarvis. *Concurrent Systems: Formal Development in CSP*. International Series in Software Engineering. McGraw-Hill International, London, UK, 1995.
25. M. G. Hinchey, J. L. Rash, and C. A. Rouff. Towards an automated development methodology for dependable systems with application to sensor networks. In *Proceedings of the IEEE Workshop on Information Assurance in Wireless Sensor Networks (WSNIA 2005), Proceedings of the International Performance Computing and Communications Conference (IPCCC-05) (Reprinted in Proceedings of Real Time in Sweden 2005 (RTiS2005), the 8th Biennial SNART Conference on Real-time Systems, 2005)*, Phoenix, AZ, April 7–9, 2005. IEEE Computer Society Press, Los Alamitos, CA.

26. C. A. R. Hoare. Communicating sequential processes. *Communications of the ACM*, 21(8):666–677, 1978.
27. C. A. R. Hoare. *Communicating Sequential Processes*. Prentice Hall International Series in Computer Science. Prentice Hall International, Englewood Cliffs, NJ, 1985.
28. F. Luna and B. Stefansson. *Economic Simulations in Swarm: Agent-Based Modelling and Object Oriented Programming*. Kluwer Academic Publishers, Dordrecht, 2000.
29. W. Michael and L. Holcombe. Mathematical models of cell biochemistry. Technical Report CS-86-4, Sheffield University, UK, 1986.
30. W. Michael and L. Holcombe. Towards a formal description of intracellular biochemical organization. Technical Report CS-86-1, Sheffield University, UK, 1986.
31. W. Michael and L. Holcombe. X-Machines as a basis for system specification. *Software Engineering*, 3(2):69–76, 1988.
32. C. W. Reynolds. Flocks, herds, and schools: A distributed behavioral model. *Computer Graphics*, 21(4):25–34, 1987.
33. C. Rouff, M. Hinchey, J. Pena, and A. Ruiz-Cortes. Using formal methods and agent-oriented software engineering for modeling NASA swarm-based systems. In *IEEE Proceedings of the Swarm Intelligence Symposium*, pp. 348–355, April 2007.
34. C. Rouff, A. Vanderbilt, M. Hinchey, W. Truszkowski, and J. Rash. Formal methods for swarm and autonomic systems. In *Proceedings of the 1st International Symposium on Leveraging Applications of Formal Methods (ISoLA)*, Cyprus, October 30–November 2, 2004.
35. C. Rouff, A. Vanderbilt, M. Hinchey, W. Truszkowski, and J. Rash. Properties of a formal method for prediction of emergent behaviors in swarm-based systems. In *Proceedings of the 2nd IEEE International Conference on Software Engineering and Formal Methods*, Beijing, China, September 2004.
36. C. A. Rouff, M. G. Hinchey, W. F. Truszkowski, and J. L. Rash. Verifying large numbers of cooperating adaptive agents. In *Proceedings of the 11th International Conference on Parallel and Distributed Systems (ICPADS-2005)*, Fukuoka, Japan, July 20–22, 2005.
37. C. A. Rouff, M. G. Hinchey, W. F. Truszkowski, and J. L. Rash. Experiences applying formal approaches in the development of swarm-based space exploration systems. *International Journal of on Software Tools for Technology Transfer. Special Issue on Formal Methods in Industry*, 8(6):587–603, 2006.
38. C. A. Rouff, J. L. Rash, and M. G. Hinchey. Experience using formal methods for specifying a multi-agent system. In *Proceedings of the Sixth IEEE International Conference on Engineering of Complex Computer Systems (ICECCS 2000)*, Tokyo, Japan, 2000. IEEE Computer Society Press, Los Alamitos, CA.
39. C. A. Rouff, W. F. Truszkowski, M. G. Hinchey, and J. L. Rash. Verification of emergent behaviors in swarm based systems. In *Proceedings of the 11th IEEE International Conference on Engineering Computer-Based Systems (ECBS), Workshop on Engineering Autonomic Systems (EASe)*, pp. 443–448, Brno, Czech Republic, May 2004. IEEE Computer Society Press, Los Alamitos, CA.
40. C. A. Rouff, W. F. Truszkowski, M. G. Hinchey, and J. L. Rash. Verification of NASA emergent systems. In *Proceedings of the 9th IEEE International Conference on Engineering of Complex Computer Systems*, Florence, Italy, April 2004. IEEE Computer Society Press, Los Alamitos, CA.

41. C. A. Rouff, W. F. Truszkowski, J. L. Rash, and M. G. Hinchey. A survey of formal methods for intelligent swarms. Technical Report TM-2005-212779, NASA Goddard Space Flight Center, Greenbelt, MD, 2005.
42. M. Savage and M. Askenazi. Arborscapes: A swarm-based multi-agent ecological disturbance model. Working paper 98-06-056, Santa Fe Institute, Santa Fe, NM, 1998.
43. S. Schneider, J. Davies, D. M. Jackson, G. M. Reed, J. Reed, and A. W. Roscoe. Timed CSP: Theory and practice. In *Proceedings of the REX, Real-Time: Theory in Practice Workshop*, pp. 640–675, volume 600 of LNCS, Springer-Verlag, June 3–7, 1991.
44. C. Suhl. RT-Z: An integration of Z and timed CSP. In *Proceedings of the 1st International Conference on Integrated Formal Methods (IFM99)*, York, UK, June 1999.
45. D. J. T. Sumpter, G. B. Blanchard, and D. S. Broomhead. Ants and agents: A process algebra approach to modelling ant colony behaviour. *Bulletin of Mathematical Biology*, 63(5):951–980, 2001.
46. W. Truszkowski, M. Hinchey, J. Rash, and C. Rouff. NASA's swarm missions: The challenge of building autonomous software. *IEEE IT Professional*, 6(5):47–52, 2004.
47. W. F. Truszkowski, L. Hallock, C. A. Rouff, J. Kerlin, J. L. Rash, M. G. Hinchey, and R. Sterritt. *Autonomous and Autonomic Systems with Applications to NASA Intelligent Spacecraft Operations and Exploration Systems*. NASA Monographs in Systems and Software Engineering. Springer-Verlag, London, UK, 2009.
48. W. F. Truszkowski, M. G. Hinchey, J. L. Rash, and C. A. Rouff. Autonomous and autonomic systems: A paradigm for future space exploration missions. *IEEE Transactions on Systems Man and Cybernetics—Part C: Applications and Reviews*, 36(3):279–291, 2006.
49. H. Van Dyke Parunak and R. S. Vanderbok. Managing emergent behaviour in distributed control systems. In *Proceedings of ISA-Tech'97*, Anaheim, CA, 1997.
50. G. Weiss, ed., *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*. MIT Press, Cambridge, MA, 1999.

第 3 部分

交通运输系统

第4章　形式化方法在铁路交通信号中的应用趋势

Alessandro Fantechi

电子信息系，佛罗伦萨大学（Università degli Studi di Firenze），佛罗伦萨，意大利；ISTI-CNR，比萨，意大利

Wan Fokkink

计算机科学系，阿姆斯特丹自由大学，阿姆斯特丹，荷兰；嵌入式系统规范和验证，CWI，阿姆斯特丹，荷兰

Angelo Morzenti

电子信息系，米兰理工大学，米兰，意大利

4.1　简介

全国铁路系统是由不自主开发基于计算机系统的运输服务供应商进行管理的。相反，只是作为从外部供应商购买的系统集成商。因此，服务供应商存在着子系统收购和管理的集成问题。所以，这种组织需要具有清晰明确的、尽可能形式化的规范要求。无论是买方还是供应商都必须同意执行严格的验收程序，根据验证和确认进行功能和安全评估，并执行安全审批机制。此外，对于允许监控系统开发过程以及便于操作和维护，统一且尽可能标准化的文件是必不可少的。根据相应的国家及国际法律，安全第一的铁路系统必须在程序设计、配置和维护上满足国际标准，这样就使得采购任务因要求严格而变得更加困难。

通常认为铁路信号形式化通常被认为是采用形式化方法的最有成效的干预领域之一。在该领域已有许多关于有关形式化规范和验证技术的成功应用的案例已经在这个领域广为流传。最近一次在 Dines Bjϕrner[7] 的一篇相关综述文章中采用了大量参考文献（仍远不完全）来表明形式化方法在该领域的应用范围，其中包括 182 篇文献以及过去两年内的引用。此外，由于保密原因，还有很多铁路公司的工作没有公布。

在铁路信号方面成功应用形式化方法的原因主要有两个方面：一方面，由于其安全重要性且无需复杂计算和硬性实时约束，使得研究人员对铁路信号的形式化方法产生浓厚兴趣，使其由此使得铁路信号成为一个具有发展前景的应用领域；另一方面，在基于简单的故障安全原则的基础上，铁路系统一直以来具有高

度的安全传统。在引入计算机之前，大多数信号系统均采用机电设备，在任何危急情况下，通过重力使得系统进入故障安全状态（如所有信号变为红色）。实际上，由于计算机没有重力，即一般而言无法预测故障所造成的影响，因此导致铁路公司迟迟不接受计算机控制的信号设备。运用非常稳定的技术并要求具有尽可能高的可靠性，是在铁路信号领域中应用计算机控制设备的关键因素。因此，认为形式化的安全证明或验证是十分必要的。

在本章中，分析了形式化方法在该领域的实际工业应用，尽管尚未满足上述成功案例的要求，但在稳步提高。接下来，还讨论了推动在工业中选择形式化方法的外部条件，并按照适用于形式化方法的程度，对铁路信号设备进行分类。在此，给出了对于该领域中形式化方法应用的一些个人的部分观点和经验，并不打算全面介绍铁路信号领域中形式化方法的应用情况。特别是，只研究了欧洲铁路信号方面，这是形式化方法在铁路领域最重要的应用场合，发现在过去10年中发生了巨大变化。

本章的结构如下：在4.2节中，讨论了欧洲电工标准化委员会（CENELEC）关于铁路信号软件发展的EN50128标准；在4.3节中，介绍了不同形式化方法在铁路信号中适用性的案例比较研究；4.4节专门研究了形式化方法，即B方法在铁路领域的应用；4.5节着重于在铁路信号设备上应用形式化方法，可分为列车控制系统和联锁系统；最后，4.6节给出了一些结论。

4.2　CENELEC标准

由CENELEC发布的EN50128标准[28]，主要是关于“铁路控制和保护系统的软件”开发，并作为铁路信号设备制造商在欧洲的主要参考依据，这些标准可扩展到其他地区和铁路行业的（其他安全相关的）其他部门。

EN50128标准是有关铁路控制和保护系统安全系列文件中的一部分，其中定义了软件安全完整性等级（SSIL）这一关键概念。这些标准表明在系统开发中，首要步骤之一是通过风险评估过程，在相关风险等级基础上，为每个组件定义一个安全完整性等级（SIL）。为不同组件分配不同的SIL，有助于集中解决关键组件（及其生产成本）。SIL分为4（非常高）、3（高）、2（中）、1（低）、0（非安全相关）级。

EN50128标准既不是一种精确的软件开发方法，也不是任何特殊的编程技术，而是采用相对于组件的SIL级别评级的技术（从“禁止”到“强烈推荐”和“强制要求”）对大量组件进行分类。强烈推荐具有较高级别SIL的系统/组件规范中采用形式化方法（尤其是CCS、CSP、HOL、LOTOS、OBJ、时态逻辑、VDM、Z和B作为示例）。同时，也强烈建议在验证活动中采用形式化方法证

明。由于也可接受其他更传统的技术，因此都未划分为强制要求。但是，值得注意的是，这是第 1 次（第 1 版 EN50128 日期追溯至 1994 年）在标准中出现形式化方法应用的明显指示。

事实上，尽管已有 CENELEC 标准和成功案例，形式化方法还未贯穿于整个铁路信号工业，大多数软件仍以传统方式编制。这是由于需要投资建立一种形式化方法的文化以及高成本的商业支持工具。此外，设备在无需应用形式化方法的条件下能够符合 CENELEC 标准。通过全面完整的测试验证可认为符合 EN50128 标准。但对于最高级的 SIL，标准要求由两个独立团队分别进行独立的设计和验证。仅依靠测试，使得测试部门中不太重要（因此投资较少）的测试工作肩负大量任务（通常超过总开发任务的 50%）。这对于市场越来越要求兼容 CENELEC 标准的公司是一种风险。唯一的解决方案是通过在规范和设计阶段引入形式化方法，将重任重新转移到设计团队。公司逐渐意识到这一必要性，且成功案例和 CENELEC 标准是提高这种意识的一个重要因素。

4.3　铁路信号系统软件采购

本节介绍了米兰理工大学和意大利国家铁路局 FS，基础设施部门（并称为 RFI，意大利铁路网络公司）之间一个联合项目的经验。该项目的目的是定义管理安全关键信号设备软件采购的流程和规则[32]。这些规则包括广泛使用的设备、控制信号和轨道跟踪、铁路/公路岔口以及列车运行。施加了附加约束条件的该项目目标如下：

- 涵盖系统开发的所有阶段，从确定需求到实施、最终验证、审核和验收；
- 提供软件开发过程中所采用的方法、语言和工具，不偏向任何特殊技术或工具供应商，唯一的总体要求是技术合理性和计算机科学中的最新研究进展；
- 提供符合或可接受的不满足国际标准（主要是 EN50128 标准，见 4.2 节）的结果；
- 选择已在工业中得到成熟应用，具有自动化工具支持，并可能获得铁路和计算机技术领域的普通工程师认可的方法。

项目的主要工作小组是由两位来自米兰理工大学从事形式化方法研究的专家以及两位来自 RFI 精通于铁路信号设备建模和分析的工程师组成的。在评估各种形式化方法时没有可以采用的正规程序或公理，而是通过对技术文档和科学文献仔细调查研究并在使用符号和工具的前人经验基础上分析而得的。为验证所得结果，并提供一些经验支持，进行了一个包括简单信号装置形式化规范的小规模实验。对各种符号中的描述，从生产的人工制品的紧凑性和可读性进行比较，这两种特性认为是有利于方法得到实际应用的最重要特性。

根据所公布的项目、需求和建议结果，定制开发不同类型的系统。该项目的主要贡献在于对需求的规范、检验和验证。特别在规范阶段，当得到合适工具和验证审核技术的支持时，建议采用形式化方法。在此，给出关于形式化需求规范的方法、工具和符号的评估比较结果。

4.3.1 系统分类

这些建议是基于系统的3种分类：复杂性（低、中、高）、关键性和时序要求。复杂性传统上是由买方确定的，而关键性是根据应用程序的SIL（见4.2节）确定的。时序要求可分为3类：时间独立、定性时间和定量时间。时间独立类是指系统没有任何特定的时间限制，例如，进行纯数据或信号描述。定性时间类是指系统可以在没有时刻和时间间隔定量信息下限制为按时间排序的值的动作。一些不怎么好的实时系统属于这一类：具有严格要求和绑定规则的系统，但设计为需充分管理每一个可能的延迟事件、预期事件或未按预期出现的事件。定量时间类别包括硬实时系统。这些系统与不完全可控的过程相互作用，并不能避免可能产生严重后果的时序约束（而不只是事件之间的顺序关系）的定量表示。需要指出的是，定性时间类系统与所谓的软实时系统完全不同，即系统中缺少一些（定性表述的定量表示）时间约束是不被期望或让人困扰的，但不会造成严重损失，而且不适用于必须具有大量数据处理能力的高吞吐量系统，但具有统计项表示的时序要求。这是因为所考虑的系统任何情况下在安全和经济方面都是非常关键的，以至于不允许没有时序需求（即使这些要求都是定性的）。

4.3.2 需求分析和规范

表4.1表明根据系统的SIL、复杂性和时序特性，给出的规范和验证技术以

表4.1 验证清单

		SIL		复杂性			时序		
		1，2	3，4	低	中	高	工业	质量	数量
分析	仿真/轨迹生成	是	是	*	*	是	*	是	是
	性能证明	无抽象	*	是（3）	*	*	*	*	*
		有抽象	*	是	*	*	*	*	*
		一般性	*	是（3）	*	*	*	*	*
		自动化程度	*	S	*	*	*	*	*
语法检查		是	是	*	是	是	*	是	是
规范覆盖度		T	T	*	*	*	*	T（1）	T（1）
验证准确度		β	γ（2）	0	α	β	0	β（1）	γ（1）

*没有推荐技术，对所用技术既不赞成也不反对。

（1）指示范围或准确度是时序部分的最低要求。

（2）在当前技术下只是推荐，并不是强制要求采用准确度为δ的方法。

（3）在当前技术下只是推荐采用，并是不强制要求。

及功能测试用例的生成。

该表分为 4 个部分：分析、语法检查、规范覆盖度和验证准确度。分析部分分为两种规范验证行为：模拟或轨迹生成以及性能证明。①系统行为的*模拟*意味着按时间顺序生成（可能以一种交互或半自动方式）事件和动作，而轨迹生成意味着产生（可能也是以一种半自动方式）系统的执行轨迹，并自动验证这些轨迹是否符合规范。与模拟不同，在轨迹生成中，事件和动作不一定按时间顺序生成。②*性能证明*表明如何在数学上证明（通过逻辑展示或穷举分析）系统具有安全、无死锁等性能。这样的证明可分为 4 类：无抽象、有抽象、一般性和自动化程度。(2.1)*无抽象*意味着可以对完整的系统规范进行证明，因此具有完全确定性：指定系统肯定具有所证明的性能。(2.2)*有抽象*意味着通过引入最初规范的适当近似（抽象）来进行证明，如忽略系统中的实际数据。抽象可使得证明更加简单，但可能会降低结果的准确性。(2.3)*一般性*意味着可以由用户以一种通用且灵活的方式，通过使用合适且充分表达的数学符号来选择将要证明的性能。(2.4)*自动化程度*是用于评价证明工具所提供的支持：S 表示至少需要一种半自动支持（用于支持证明正确的验证支持，例如，通过准备证明义务结构或自动证明普通组件和子证明来实现，但必须由专业用户执行），否则可手动证明。

*语法检查*用于表示是否存在检查规范语法正确性的工具支持。对于规范覆盖度，T 表示完全覆盖：所有要求都具有相同的相关性，因此必须进行规范。否则可能出现的情况是：以一种明确方式确定且与其他要求完全隔离的一些要求对安全没有任何影响，且无需形式化规范。最后，*验证准确度*测量验证要求规范的准确度。为提高准确度，可选择的值为：0（非正式检查，预演）；α（类型的语法控制，定义之间的一致性和组成规范的实体使用，即由现代编程语言的编译器进行典型的静态控制）；β（至少下列之一：仿真、模拟、轨迹生成、符号分析、可达性分析，无死锁等特定性能证明、抽象性能证明）；γ（与 β 相同，但至少结合两个不同性质的技术，并采用合适的尺度来衡量系统分析的覆盖度），δ（一般性能的表述和证明）。

值得注意的是，表 4.1 中的列分别表示不同系统功能，但所列方法应结合起来应用（在其他项中，采用更苛刻的要求）：例如，对于一个中等复杂度并具有定性时序特征的系统，所需的验证准确度为 β；对于一个复杂度较低但非常关键（SIL 为 3 或 4）的系统，所需的验证准确度为 γ。

最后，表 4.2 表明表 4.1 中所列方法应用到在当时项目中广泛采用的一致形式，且同时考虑两种语言特点和可用的支持工具。分析符号和形式化方法有 Z[46]、TRIO[34]、Statecharts[36]、SDL[23]、UML[14]、LOTOS[48]、Petri nets[45]、SCADE[15] 和 B[1]。通过对每个符号、相应方法和工具环境，比较表 4.1 中要求表

表 4.2 规范方法清单

	SIL		Complexity			Time		
	1，2	3，4	Low	Med	High	Ind	Qual	Quant
Z	Yes	No①	Yes	Yes	No①	Yes	Yes	No
TRIO	No①	No①	Yes	No①	No①	Yes	Yes	Yes
Statecharts	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes②
SDL	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes②
UML	No③	No	No③	No	No	Yes	Yes	No
PN	Yes	No	Yes	No	No	Yes	Yes	Yes②
LOTOS	Yes	No	Yes	Yes	No	Yes	Yes	Yes②
SCADE	Yes	No①	Yes	Yes	Yes	Yes	Yes	Yes②
B	Yes	Yes	Yes	Yes	Yes	Yes	Yes	No

①“NO”是由于得不到具有需要“Yes”值的所有功能的完全综合水平的工具。

② 对于 SIL 为 1 或 2 级的系统推荐采用该方法，而对于 SIL 为 3 或 4 级的系统，在目前技术下可采用该方法，但不强烈推荐。

③ 对于 SIL 为 1 级的系统，只推荐该方法。

示的典型特征和支持工具可得到表 4.2。毫不奇怪，在定量定时和高度复杂的系统情况下，目前技术并不理想，即使是已取得最好成绩的方法和工具。在这种情况下，没有“强烈推荐”的方法和工具，现有的方法和工具中只有“建议”。这是由于当前可用工具在分析和验证形式化模型时缺乏严谨的形式化基础，或没有在工业环境下重复且持久的应用中得到认证和验证。然而，正如在原书前言中所述，在铁路信号中一般没有复杂计算或硬实时约束。

4.4 成功案例：B 方法

从 4.3 节的比较案例研究中可知，B 方法[1]是功能最强的验证方法之一。B 方法具有严谨的数学基础和成熟的基本方法，且得到非常先进的工具集的支持。一系列的铁路信号产品得益于在其设计过程中应用 B 方法。通过影响 EN50128 标准的定义（见 4.2 节），B 方法的成功在铁路信号领域有着重大影响。

B 方法的目标是在软件开发中从规范化、精细化到实现和代码自动生成，验证每个阶段。这包括一个规范系统的符号—抽象机：抽象机定义为可修改状态变量值的一组状态和一组操作；定义状态的一个不变谓词；对于每个操作，定义一个描述操作对状态变量影响的前提条件和后置条件。必须证明在同时满足前提条件和不变量的状态中执行一个操作时，操作执行后的状态满足后置条件和不变量。而且，在每个细化阶段，必须证明保留了系统所需的安全性能。所以定制规

范可产生一系列需由形式化证明执行的证明义务。与 B 方法一起，还具有相应的支持工具，其中包括用于推导证明义务、定理证明和 Ada 代码生成的工具。

B 方法已在铁路信号系统得到成功应用，特别是主要在法国的 Matra 交通运输和 Alstom 中。第一次应用于 20 世纪 80 年代末期巴黎 RER 铁路交通线控制的 SACEM 系统[20]。为保证两家铁路公司能够对铁路交通线（SNCF 和 RATP）进行正确设计，在已开工的项目中引入了 B 方法。后来，在由同一公司设计的类似系统中均采用 B 方法（特别是现已被西门子公司收购的 Matra 公司）。最引人瞩目的应用之一是自从 1998 年 10 月开始运行的巴黎 Meteor 地铁线。这条地铁线设计为每小时达到 4 万乘客的运力，且在客流高峰期列车之间的间隙可降为 85s。它通过 Matra 公司开发的、包括 86000 条 Ada 地铁线的列车自动操作系统进行管理（见文献［3］）。

在 B 方法的开发过程中证明了 27800 条定理，其中，大约 90% 是通过支持工具自动证明，其余的约 2000 条定理是通过交互式证明的。在证明过程中发现许多错误。相比之下，B 方法开发后，各种测试过程中都没有检测到进一步的错误。此外，自从地铁线运行后，也没有错误报告。

4.5　铁路信号设备分类

铁路信号装备大致可分为两大类：列车控制系统为列车的安全行驶速度和制动控制提供保障，而联锁系统通过错综复杂的轨迹和站点布局建立安全行驶路径。另外，还需考虑其他几个常用来为主信号系统提供输入的小信号系统，其中一些小信号系统与主信号系统具有相同的关键等级。另外，某些信号系统实际上整合了上述两类系统的特点。然而，为分析起见，在此仅主要考虑两类主要系统的特性。

4.5.1　列车控制系统

目前，已有各种各样的列车控制系统，这可能取决于司机对车辆的控制程度（从几乎不提供任何支持的手动系统到完全自动化的无人驾驶系统），以及取决于列车信息传送的不同方式（司机或车载设备）和信息性质。然而，最基本的列车速度控制原则是共同的：*制动曲线*的概念（见图 4.1）。根据前方列车或固定障碍，可定义铁路线上任一点处给定时刻的列车最大安全速度曲线。必须保证列车速度低于曲线上的相应速度。由于前方列车不断运动，曲线也随之变化，提供列车的安全车头时距。面临的主要挑战是要确保列车的车载系统曲线足够准确。

当在车载系统和沿线站点计算机之间不断交换信息时，要求具有带宽保证且

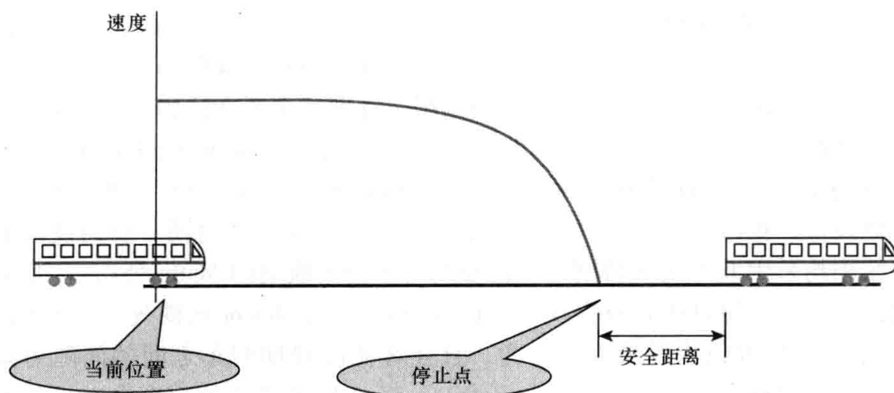


图 4.1 制动曲线原理

应采用安全通信协议。因此，在现代列车控制系统中，是从基本安全规则复杂性转变为通信协议的复杂性。

从 SACEM 和 Météor 系统中可以看出形式化方法可应用到整个系统开发过程。然而，在处理主要线路时，由于异质性、兼容性以及互操作性问题，使得情况变得复杂。机车通常要求在货运/客运混合的不同装备线路上运行。每个国家的铁路都有其自身的传统，尤其是涉及信号规则和程序方面。在开放的欧洲市场出现之前，各国铁路公司都与信号设备供应商（大多为国家公司）保持联系，以及在典型的受保护市场下，铁路公司和供应商都采用一种全国范围的特定信号方式来保持严格关系。这使得欧洲各国有着需求不同车载设备的不同列车控制系统。而开放性铁路市场的新规则要求任何一家公司生产的列车可行使在欧洲所有铁路线上。这意味着列车应具有互操作性。实现该目标的唯一方法就是在列车驾驶室内安装所经国家的多个不同版本的车载系统，或在边境处更换机车车头。

出于上述原因，设立欧洲铁路交通管理系统/欧洲列车控制系统（ERTMS/ETCS）项目，旨在实现未来欧洲铁路网的单一列车控制系统[25]。在目前的初始测试阶段之后，该项目计划逐步在传统设备上并排安装 ERTMS/ETCS 设备，同时根据铁路沿线站点计算机与车载设备之间信息量的增加程度来开发 3 种连续的 ERTMS/ETCS 等级。在等级 2 和等级 3 中，利用 GSM-R（专用于铁路行业的 GSM 无线通信）来为铁路线上的前方列车连续传输信息。

ERTMS/ETCS 充分利用标准化组件（欧洲关键板载计算机、无线闭塞中心和应答器）和由欧洲主要信号制造商根据综合技术联合提出的协议（欧洲电台）。ERTMS/ETCS[27] 所发布的规范是一种包括表格、状态图和序列图以增加形式化程度的结构化自然编程语言需求文档。

目前，已开展了一些关于 ETCS 协议和组件形式化建模和验证的研究，从有

色 Petri 网[43]提供的 ETCS 模型开始，到通过利用状态图来证明早期基于无线电的列车控制系统[19]模型校验的安全性能，到通过随机 Petri 网[49]或 CSP-OZ-DC[29]研究实时性能。实现 ETCS 自然语言需求形式化的最系统化方法已在最近由欧洲铁路局资助的 EuRailCheck 项目[16]中进行研究，其中开发一种在受控自然语言约束条件下的 UML 增强图来产生形式化需求段。通过一种定制的 NuSMV 模型校验器可实现这些需求段的自动验证分析。

形式化方法还用于联盟公司的开发周期中。例如 Ansaldo 公司，在利用 SDL 和消息序列图解决无线闭塞中心建模和检验[17]后，通过 UML 状态图方式给出一种欧洲无线通信协议的形式化规范，这样就可在将场景 UML 序列图表示之后进行模拟检验[26]。

在 ETCS 情况下，形式化方法的研究热点已经从列车控制综合逻辑（制动曲线原理）转变为基于无线的安全性能和实时性能，这是即使信号不再出现在铁路线上，轨道前方信息仍可与列车通信的唯一一种方式。

4.5.2 联锁系统

铁路领域的控制和管理包括两个独立任务：第一，轨道和站点的控制指令是在通常由专家管理的逻辑层上设计的；第二，必须保证控制指令的执行不会危害到列车安全，即必须避免碰撞和出轨。这可由称为联锁系统的方式完成，这是介于基础设施和逻辑层及其接口之间的媒介。

联锁系统是一种嵌入式系统，用于控制设备部件（如信号装置、站点装置、轨道电路和自动闭塞装置）之间相互连接，使其能够按顺序执行功能，并为保证操作安全而定义联锁系统。一个简单的联锁系统如图 4.2 所示，这是来自一个实际的意大利联锁系统[21]。图中，线段表示基础设施中的轨道；其中有的含有轨道电路，即用圆圈内的标号表示列车中的传感器；轨道段之间的交点表示站点。棒棒糖式的图形表示各种不同类型的信号。在整个轨道路线中的每个重要部分都有数字标记。该示例（只有 1 条铁轨的车站）是由 8 条可行路径、2 个站点、8 个信号装置、6 个轨道电路装置和 2 个自动闭塞装置组成。

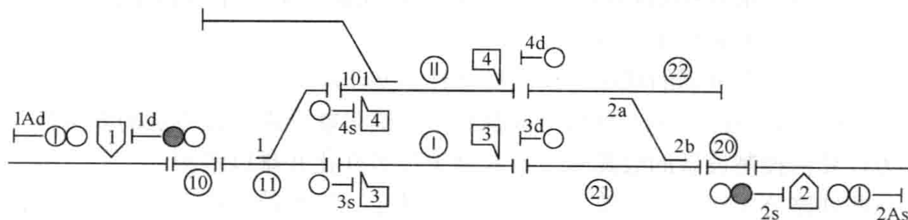


图 4.2 最简单的路线布局

只有当正确配置轨道上所有站点，且无列车时才能设置该轨道为安全。同时，只有前方轨道设置为安全，才能将信号设为绿色。上述表述是所有联锁系统必须遵守的两条一般规则。这些规则旨在仅允许站点位置、信号等安全组合，以避免碰撞。由联锁系统处理的信号指示装置控制着正确使用轨道并授权在其上的列车运动。通常，这些规则会执行一个预定义的动作序列。例如，发布一个轨道请求指令，首先会触发检查该线路上的所有轨道装置是否正确设置。在此情况下，针对线路上站点位置以及闭锁轨道装置而发布指令。在此之后紧接着是一个对受控装置的正确状态进行全局集中控制的阶段，然后就可以闭锁该线路，并设置相应的信号指示装置。

然而，需要注意的是，上述表示的一般规则需要与每种特定布局相结合。例如，对于图 4.2 中的线路 1-3，应考虑站点 1，轨道电路 10、11、I 等来设置规则。

实际上，这些规则的精确完整的设置取决于轨道与站点的布局（参见文献[30]）以及铁路公司或监管机构发布的传统国家政策。由于联锁系统是安全性的关键所在，因此形式化这些规则是首要要求。

许多铁路公司在开发基于继电器的联锁系统的传统工艺过程中，将一般原则编码到继电器电路模板中（继电器是一种电动开关，使得电流通过铁心周围缠绕的铜线，产生吸合横杆的电磁场，从而改变开关触点）。在安装一个新的联锁系统时，必须使得这些一般原则适用于轨道的特定布局。对于某些形式化程度不一的规则也应遵循该适用过程。上述过程结束时，会产生一个包含车站中每个逻辑或物理对象的命令和控制电路的示意图。

一种该类型的示意图如图 4.3 所示；该示意图来自同一个意大利联锁系统，并表示图 4.2 中考虑站点 1 以及轨道电路 10、11、I 等时线路 1-3 的电路。图 4.3 中的梯形图表明继电器 CD1_3 的通电取决于许多其他的继电器触点。

这种电路是基于有助于工程师新车站设计的模板而生成的。然后，将该模板和布局目标相关联，并复制每个目标。在一个电路模板中，包含所有需要对这些目标进行管理的触点。执行的唯一动作是利用布局显示的触点来代替传统触点。线路相关的所有示意图结构都相同，但从一条线路到另一条线路上的触点（串联或并联）编号和名称都不相同。

过去，基于继电器的联锁系统的安全性是基于单个故障安全的概念，并利用中继技术的固有特性。引入计算机控制和指令链可能会影响该方法的安全性，这是因为计算机控制设备的故障模式可能会更加多样化并难以预测。

一些铁路公司所采用的第一种方法是在基于计算机的联锁系统开始中，保持传统且成熟的基于继电器的原理图作为值得信赖的信息来源，然后通过成本较高且繁琐但并非穷举的测试手段，以寻求新的联锁系统与信息源的一致性。该方法

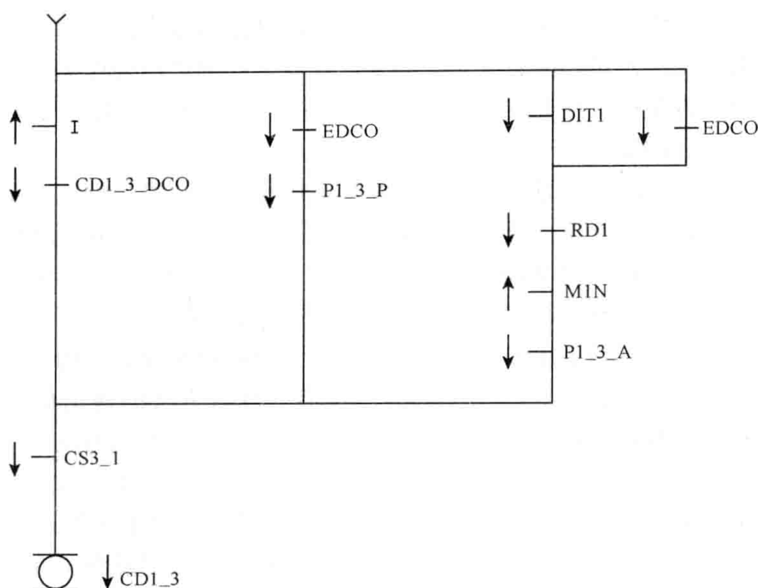


图 4.3　线路 1-3 的瞬时继电器方案

对供应商研发联锁系列产品施加了压力，并结合通过考虑某些性能来实例化通用产品的方式，以将已证明与可信信息来源兼容的适合的形式化原理图解释或编译到运行代码中。从文献［6，24，33，42］可知，不同制造商所采用的实际方法也各不相同。由于能够明确规范保证线路安全设置的逻辑规则，因此开发采用形式化方法的计算机控制联锁系统得到越来越多的兴趣和关注。B 方法的符号（见 4.4 节）并不真正适合于联锁系统的逻辑规则表示，这是因为许多逻辑规则需要全局访问系统状态（对应于原来继电器状态的逻辑变量），而抽象机对这些状态变量进行封装，使得只能通过操作才能访问。

如今，不再是依靠原先的中继技术，而是需要一种对整个系统进行完全形式化的创新方法。现已开发出用于联锁系统形式化的特定域编程语言[37,44]，其中最著名的是欧洲铁路联锁规范（EURIS）[4,47]，这将在 4.5.3 节中详细讨论。目前，使用可能商业化的支持工具已大大推进通用编程语言的应用，如 4.3 节所述。并且，最近的发展趋势表明状态图（其中的 Statemate、Stateflow 或 UML 状态图语言）可作为一种定义标准形式化的方法，例如，参见文献［2］。

状态图的发展趋势已从传统受保护的市场向欧洲开放市场过渡，只不过是以不同的理由而非针对列车控制系统（见 4.5.1 节）。因为在以前的国家市场保护政策下，由国家制造商为国家铁路公司开发联锁系统。铁路信号仍然是负责铁路基础设施的政府部门职责（如 RFF、RFI、ProRail），而不是开放市场的铁路运营商的责任。另一方面，传统的国家工业已由少数跨国公司合并和重组。因此，

为统一产品生产线，这些跨国公司必须整合不同铁路信号产品的相关技术。在此情况下，国家铁路公司和国家铁路产品生产企业之间的严格合作关系由于不需要明确规范而不再存在（所有误解都可通过电话解决）。这就更加需要明确定义涉及各方责任的合同规范。

欧洲铁路部门已逐级意识到对当前联锁系统设计必须进行彻底改革，这主要有4个方面的原因。首先，目前联锁系统的设计方法，如Alstom的SSI和VPI以及Westinghouse Signals的WESTRACE等方法，都是基于继电器的联锁早期设计方法，因此没有充分利用计算机硬件和软件的附加功能。其次，由于缺乏模块化，目前的方法不适用于建设超大规模铁路车站的联锁系统；采用最常见的解决方案，即将该车站分为独立的几个部分，会造成现有方法的通信成本过高。第三，由于缺乏标准化，到目前为止，不同欧洲国家仍采用不同的联锁技术，导致成本较高。第四，形式化方法难以集成到现有方法中，如参见文献［18，22，35］中有关这方面的一些研究工作，主要是采用用于联锁系统形式化验证的模型校验技术。正是由于上述原因，意大利铁路基础设施部门发布了在4.3节中讨论的软件采购文件。

欧洲联锁系统开发项目促使欧洲主要铁路公司和供应商开发联锁系统的欧洲功能要求和标准化接口。在这种情况下，由于联锁系统不与列车（直接）通信，互操作性并不是一个主要问题。标准化的唯一目的是通过标准化组件和标准化联锁规则来降低成本。在欧洲联锁系统中，EIFFRA研究团队[40]着重于文本需求、需求管理工具，如Telelogic的DOORS，以及基于模型的要求。这是通过利用UML状态图和Statecharts来描述行为，以及利用OCL来描述联锁系统的性能而实现的。在此背景下，SNCF-RFF利用Statemate对（基于继电器的）联锁系统原理图建模，生成联锁装置的90个通用Statecharts[41]。该方法已在一个中等规模的车站中实例化，获得25个Statecharts（90个之中）的115个实例。利用Statemate的仿真以及Waveforms的可视化场景来验证定义和实例的正确性。

近期提出的一个欧洲联锁系统项目是FP7 INESS项目[38]，目的是定义新一代联锁系统的规范，尤其是注重与ERTMS系统的正确接口。在INESS中，已选用UML状态图作为建模语言。

4.5.3　EURIS语言

联锁逻辑对系统状态的各种限制完全一致，主要是取决于自主装置的参数，如信号和站点。根据这一现象，来自荷兰ProRail公司的Peter Middelraad开发了一种模块化规范方法/语言EURIS[4]，来描述完全自动化的联锁逻辑。EURIS假定了一种面向对象的体系结构，其中包括一组通用结构块用于表示信号和站点等

基础设施中的装置，以及两个与外界完全隔离的实体来表示逻辑层和基础设施。在这些共同组成联锁逻辑的结构块之间通过称为报文的消息相互通信，同时，这些结构块还可以与两个建模为逻辑层和基础设施的实体交换报文信息。该模型的描述如图 4.4 所示。

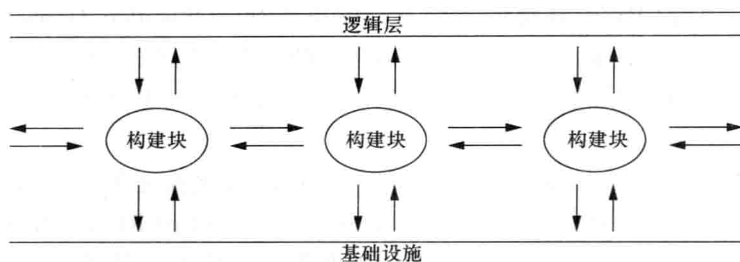


图 4.4　EURIS 架构

在此，给出一个示例，假设逻辑层决定列车应通过线路 R 运行。将该请求传送到声明占用线路 R 的联锁层。在此，可将该任务分为多个通过结构块之间交换报文信息来完成的小任务。如果所有结构块都一致同意线路 R 成立且不会危害安全性，那么联锁系统将占用这条线路，并对基础设施发送必要指令。

EURIS 不仅表示一种规范方法，同时也命名了一种基于该方法的图形化面向命令规范语言。利用一个所谓的逻辑和序列图（LSC）来指定一个结构块。每个包含程序图形化表示的 LSC 能适应和测试变量值并最终触发报文发送。该报文可由相邻结构块、重叠逻辑层和底层基础设施接收。反之，每个结构块也可从相邻结构块、逻辑层和底层基础设施接收报文。这种在结构块、逻辑层和基础设施之间的通信通道以及变量初始值都保存在 LSC 外部。

在 EURIS 中，规范的核心是定义了各种不同结构块处理输入报文的方式。直观地说，这些结构块代表基础设施中的独立装置，如信号和接点。只要所有类型的结构块都已详细说明，那么特定铁路区域的布局规范可以适当方式简单连接分离的结构块来构建实现。面向对象的 EURIS 方法对于大规模铁路车站的联锁系统设计方面不会产生额外成本，并对于不同国家可能有不同的联锁设计模式。

UniSpec[4]是 EURIS 方法的一个特殊实例，是由 ProRail 公司开发的组成联锁系统的一整套通用装置。其中，一个模拟器可使得 UniSpec 规范行为生动形象。设计完一组 LSC 后，用户可以根据铁路拓扑结构来添加 LSC 实例。对设计规则的错误和编译结果进行检查，然后，可通过图形化界面来模拟仿真受控铁路区域的情况。

在由荷兰 Holland Railconsult 公司资助的一个项目中，来自 Utrecht 大学的研究人员为 EURIS 制定了形式化操作语义[5]，这是根据 EURIS 模拟器，基于离散

时间模型的。该语义用于离散时间进程代数的设定。在由 ProRail 资助的后续项目中，来自阿姆斯特丹工人国际委员会（CWI）的研究人员设计了一种名为 LARIS 的文本形式 EURIS[31]，旨在提高图形化 LSC 的清晰度。EURIS 规范的验证工作正在阿姆斯特丹的 CWI 进行。另外，还实现了 EURIS 到 μCRL[8] 的编译器原型，并在 μCRL 工具集的帮助下，解决了荷兰 Woerden-Harmelen 车站的 EURIS 规范验证。因此，可提供该联锁系统的状态空间符号版本用于分析。利用 μCRL 工具集可实现一个（更小的）虚构火车站的 EURIS 规范的正确性验证。

有关 Woerden-Harmelen 的验证工作使得在形式化验证领域取得一些进展。即由于该车站联锁系统的状态空间太大，因此必须利用 μCRL 工具集来实现新的验证方法，以应付如此大的状态空间。这些新的验证方法基于部分降阶法[12]、分布式状态空间生成法[13]以及分布式状态空间最小化[9-11]。

EURIS 的所有权已经从 ProRail 转卖给西门子公司，以保证更加强大的商业支持和工具开发。目前，EURIS 是西门子公司 GRACE 工具集的核心[39]。

4.6　小结

随着欧洲共同体强制推行开放市场，铁路发展在过去的几十年内发生了巨大变化，但可以看到形式化方法在铁路信号中的应用发展历史并未脱离铁路的发展历史。其中，见证了向基于行为、状态机的形式化转变，以及商业化工具越来越多地关注向形式化支持的转变。具有仿真和模型校验规范并生成代码的工具将会产生附加值。但是，这对于主要从事软件生产或微型设备的小公司来说，可能由于太过于昂贵而望而却步。在得到一个令人满意的解决方案之前，还将有一段很长的时间。

规范和验证的形式化方法正逐步平稳地得到工业环境下的认可和应用：在学术界已有许多符号、方法和（原型）工具，然而，在工具稳定性、规范文档和用户支持方面尚缺乏工业实际应用，另一方面，在工业中也出现一些具有技术含量的方法和工具。尽管在实际应用中利用这些（得到工业应用）工具来实现复杂硬实时系统的全面验证仍是不可行的，但是验证技术正得到快速发展。

EN50128 等国际标准在推进采用系统和技术完善的开发方法中具有积极作用，但也存在技术过时、晦涩难懂、模棱两可或过于迁就等问题。近期，TC9X/SC9XA 技术委员会已公布建议修订 CENELEC 规范，其中包括更多关于形式化建模和形式化验证技术的参考。

参考文献

1. J. R. Abrial. *The B-Book*. Cambridge University Press, 1996.
2. M. Banci and A. Fantechi. Geographical vs. functional modelling by statecharts of interlocking systems. In *Proceedings 9th Workshop on Formal Methods for Industrial Critical Systems (FMICS'04), Linz, Volume 133 of Electronic Notes in Computer Science*, pp. 3–19. Elsevier, 2005.
3. P. Behm, P. Benoit, A. Faivre, and J. M. Meynadier. Météor: A successful application of B in a large project. In *Proceedings World Congress on Formal Methods in the Development of Computing Systems (FM'99)*, pp. 369–387, Volume 1708 of Lecture Notes in Computer Science, Springer, Toulouse, 1999.
4. J. Berger, P. Middelraad, and A. J. Smith. EURIS, European railway interlocking specification. In *Proceedings IRSE'93*, pp. 70–82, Institution of Railway Signal Engineers, 1993.
5. J. A. Bergstra, W. J. Fokkink, W. M. T. Mennen, and S. F. M. van Vlijmen. Railway Logic via EURIS. Quaestiones Infinitae XXII, Zeno Institute of Philosophy, Utrecht, 1997 (in Dutch).
6. C. Bernardeschi, A. Fantechi, S. Gnesi, S. Larosa, G. Mongardi, and D. Romano. A formal verification environment for railway signaling system design. *Formal Methods in System Design*, 12(2):139–161, 1998.
7. D. Bjørner. New results and trends in formal techniques and tools for the development of software for transportation systems—A review. In *Proceedings 4th Symposium on Formal Methods for Railway Operation and Control Systems (FORMS'03)*, L'Harmattan Hongrie, Budapest, 2003.
8. S. C. C. Blom, W. J. Fokkink, J. F. Groote, I. A. van Langevelde, B. Lisser, and J. C. van de Pol. μCRL: A toolset for analysing algebraic specifications. In *Proceedings 13th Conference on Computer Aided Verification (CAV'01)*, pp. 250–254, Volume 2102 of Lecture Notes in Computer Science, Springer, Paris, 2001.
9. S. C. C. Blom and S. Orzan. A distributed algorithm for strong bisimulation reduction of state spaces. In *Proceedings 1st Workshop on Parallel and Distributed Model Checking (PDMC'02), Brno, Volume 68(4) of Electronic Notes in Theoretical Computer Science*, Elsevier, 2002.
10. S. C. C. Blom and S. Orzan. Distributed state space minimization. In *Proceedings 8th Workshop on Formal Methods for Industrial Critical Systems (FMICS'03), Trondheim, Volume 80 of Electronic Notes in Theoretical Computer Science*, Elsevier, 2003.
11. S. C. C. Blom and S. Orzan. Distributed branching bisimulation reduction of state spaces. In *Proceedings 2nd Workshop on Parallel and Distributed Model Checking (PDMC'03), Boulder, Volume 89 of Electronic Notes in Theoretical Computer Science*, Elsevier, 2003.
12. S. C. C. Blom and J. C. van de Pol. State space reduction by proving confluence. In *Proceedings 14th Conference on Computer Aided Verification (CAV'02)*, pp. 596–609, Volume 2404 of Lecture Notes in Computer Science, Springer, Copenhagen, 2002.

13. S. C. C. Blom, I. A. van Langevelde, and B. Lisser. Compressed and distributed file formats for labeled transition systems. In *Proceedings 2nd Workshop on Parallel and Distributed Model Checking (PDMC'03), Boulder, Volume 89 of Electronic Notes in Theoretical Computer Science*, Elsevier, 2003.

14. G. Booch, J. Rumbaugh, and I. Jacobson. *The Unified Modeling Language User Guide*. Addison-Wesley, 1999.

15. F. Boussinot and R. de Simone. The Esterel language. Another look at real time programming. *Proceedings of the IEEE*, 79(9):1293–1304, 1991.

16. R. Cavada, A. Cimatti, A. Mariotti, C. Mattarei, A. Micheli, S. Mover, M. Pensallorto, M. Roveri, A. Susi, and S. Tonetta. EuRailCheck: Tool support for requirements validation. In *ASE 2009*, Auckland, New Zealand, November 16–20, 2009.

17. A. Chiappini, A. Cimatti, C. Porzia, G. Rotondo, R. Sebastiani, P. Traverso, and A. Villafiorita. Formal specification and development of a safety-critical train management system. In *Proceedings 18th Conference on Computer Safety, Reliability and Security (SAFECOMP'99)*, pp. 410–419, Volume 1698 of Lecture Notes in Computer Science 1698, Springer, Toulouse, 1999.

18. A. Cimatti, F. Giunchiglia, G. Mongardi, D. Romano, F. Torielli, and P. Traverso. Formal verification of a railway interlocking system using model checking. *Formal Aspects of Computing*, 10(4):361–380, 1998.

19. W. Damm and J. Klose. Verification of a radio-based signaling system using the STATEMATE verification environment. *Formal Methods in System Design*, 19(2):121–141, 2001.

20. C. DaSilva, B. Dehbonei, and F. Mejia. Formal specification in the development of industrial applications: Subway speed control system. In *Proceedings 5th IFIP Conference on Formal Description Techniques for Distributed Systems and Communication Protocols (FORTE'92)*, pp. 199–213, Perros-Guirec, North-Holland, 1993.

21. P. E. Debarbieri, F. Valdambrini, and E. Antonelli. A.C.E.I. Telecomandati per linee a semplice binario, schemi I0/19. CIFI Collana di testi per la preparazione agli esami di abilitazione, Quaderno 12, 1987.

22. C. Eisner. Using symbolic CTL model checking to verify the railway stations of Hoorn-Kersenboogerd and Heerhugowaard. *Software Tools for Technology Transfer*, 4(1):107–124, 2002.

23. J. Ellsberger, D. Hogrefe, and A. Sarma. *SDL—Formal Object-oriented Language for Communicating Systems*. Prentice Hall, 1997.

24. L. H. Eriksson. Specifying railway interlocking requirements for practical use. In *SAFECOMP'96—Proceedings of the 15th International Conference on Computer Safety, Reliability and Security*, Springer-Verlag, 1996.

25. UNIFE, ERTMS, 2012, Available at: http://www.ertms-online.com.

26. R. Esposito, A. Lazzaro, P. Marmo, and A. Sanseviero. Formal verification of ERTMS Euroradio safety critical protocol. In *Proceedings 4th Symposium on Formal Methods for Railway Operation and Control Systems (FORMS'03)*, L'Harmattan Hongrie, Budapest, 2003.

27. European Railway Agency, ERTMS, List of Mandatory Specifications and Standards, 2012. Available at: http://www.era.europa.eu/Core-Activities/ERTMS/Pages/List-Of-Mandatory-Specifications-and-Standards.aspx.

28. European Committee for Electrotechnical Standardization. EN 50128, Railway Applications Communications, Signaling and Processing Systems Software for Railway Control and Protection Systems, 2001.

29. J. Faber. Verifying real-time aspects of the European train control system. In *Proceedings 17th Nordic Workshop on Programming Theory (NWPT'05)*, Copenhagen, 2005.

30. W. J. Fokkink. Safety criteria for the vital processor interlocking at Hoorn-Kersenboogerd. In *Proceedings 5th Conference on Computers in Railways (COMPRAIL'96)*, pp. 101–110, Computational Mechanics Publications, Berlin, 1996.

31. W. J. Fokkink, J. F. Groote, M. J. Hollenberg, and S. F. M. van Vlijmen. LARIS 1.0: Language for Railway Interlocking Specifications. CWI Publications Miscellaneous, Stichting Mathematisch Centrum, 2000.

32. U. Foschi, M. Giuliani, A. Morzenti, M. Pradella, and P. San Pietro. The role of formal methods in software procurement for the railway transportation industry. In *Proceedings 4th Symposium on Formal Methods for Railway Operation and Control Systems (FORMS'03)*, L'Harmattan Hongrie, Budapest, 2003.

33. B. Fringuelli, E. Lamma, P. Mello, and G. Santocchia. Knowledge-based technology for controlling railway stations. *IEEE Intelligent Systems*, 7(6):45–52, 1992.

34. C. Ghezzi, D. Mandrioli, and A. Morzenti. TRIO: A logic language for executable specifications of real-time systems. *Journal of Systems and Software*, 12(2):107–123, 1990.

35. J. F. Groote, J. W. C. Koorn, and S. F. M. van Vlijmen. The safety guaranteeing system at station Hoorn-Kersenboogerd. In *Proceedings 10th IEEE Conference on Computer Assurance (COMPASS'95)*, pp. 131–150, IEEE Computer Society Press, Gaithersburg, 1995.

36. D. Harel. Statecharts: A visual formalism for complex systems. *Science of Computer Programming*, 8(3):231–274, 1987.

37. A. E. Haxthausen and J. Peleska. Generation of executable railway control components from domain-specific descriptions. In *Proceedings 4th Symposium on Formal Methods for Railway Operation and Control Systems (FORMS'03)*, pp. 83–90, L'Harmattan Hongrie, Budapest, 2003.

38. INESS project, 2009. Available at: http://projects.uic.asso.fr/

39. B. Jung. Die methode und werkzeuge GRACE. In *Formale Techniken für die Eisenbahn-sicherung (FORMS'00)*, Fortschritt-Berichte VDI, Reihe 12, Nr. 441, VDI Verlag, 2000.

40. N. H. König and S. Einer. The Euro-Interlocking formalized functional requirements approach (EIFFRA). In *Proceedings 4th Symposium on Formal Methods for Railway Operation and Control Systems (FORMS'03)*, L'Harmattan Hongrie, Budapest, 2003.

41. P. Le Bouar. Interlocking SNCF functional requirements description. Euro-Interlocking Project, Paris, May 2003.

42. G. LeGoff. Using synchronous languages for interlocking. In *First International Conference on Computer Application in Transportation Systems*, 1996.

43. M. Meyer zu Hörste and E. Schnieder. Formal modelling and simulation of train control systems using Petri nets. In *Proceedings World Congress on Formal Methods*

in the Development of Computing Systems (FM'99), p. 1867, Volume 1709 of Lecture Notes in Computer Science, Springer, Toulouse, 1999.

44. M. J. Morley. Safety in railway signalling data: A behavioural analysis. In *Proceedings 6th Workshop on Higher Order Logic Theorem Proving and its Applications (HUG'93), Vancouver, Volume 740 of Lecture Notes in Computer Science*, pp. 464–474. Springer, 1993.
45. T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, 1989.
46. J. M. Spivey. *Introducing Z: A Specification Language and Its Formal Semantics*. Cambridge University Press, 1998.
47. F. J. van Dijk, W. J. Fokkink, G. P. Kolk, P. H. J. van de Ven, and S. F. M. van Vlijmen. EURIS, a specification method for distributed interlockings. In *Proceedings 17th Conference on Computer Safety, Reliability and Security (SAFECOMP'98)*, pp. 296–305, Volume 1516 of Lecture Notes in Computer Science, Springer, Heidelberg, 1998.
48. P. van Eijk, C. A. Vissers, and M. Diaz. *The Formal Description Technique LOTOS*. Elsevier, 1989.
49. A. Zimmermann and G. Hommel. Towards modeling and evaluation of ETCS real-time communication and operation. *Journal of Systems and Software*, 77(1):47–54, 2005.

第5章　航空电子设备的符号模型校验

Radu I. Siminiceanu
美国国家航空航天研究所，弗吉尼亚州汉普顿
Gianfranco Ciardo
计算机科学与工程学系，加州大学，加利福尼亚州

5.1　简介

自20世纪70年代以来，地面上的机载作战管理系统和空中交通运营管理越来越自动化。商用和军用飞机驾驶舱如今已高度计算机化。这类设备通称为航空电子设备。这包括数字电子设备、零部件以及应用于以下的整个子系统：

- 导航，如全球定位系统（GPS）；
- 控制，如自动驾驶系统；
- 通信，如基于GPS的广播式自动相关监视系统（ADS-B）；
- 冲突检测和消解（CD&R）协议，如交通防撞系统（TCAS）和跑道入侵预防系统（RIPS）；
- 飞行数据记录仪；
- 集成显示系统；
- 完整飞机管理系统，如模块化集成航空电子系统（IMA）；
- 无人驾驶系统［无人机（UAS）］。

对于安全关键航空电子设备，监管部门［美国联邦航空管理局（FAA）、民用航空管理局（CAA）或国防部（DoD）］需要软件开发标准，如用于军用系统的MIL-STD-2167和用于民用飞机的RTCA DO-178B。这对于相关技术最初未包含在条例规定中的形式化方法是一个机遇和挑战，目前，该方法已取得巨大进步，并强烈建议应用于构建可靠性系统的过程中[39]。

验证航空电子系统所面临的挑战可归结为两个主要因素：

- 嵌入式系统的混合特性：无论是离散变量还是连续变量都需要如实描述模型概念，如（连续）飞机运动轨迹、操作环境中的物理量（如温度、压力、速度和加速度）以及最显著的“实时性”。
- 系统复杂度：元件数量、尺寸大小以及这些元件之间的相互作用是相当

大的。

*混合动力系统*的分析已经进行了近20年。但是，正如文献［43］中指出，研究界从一开始就采用一种基于接受“数据变换而并非物理动力学”计算理论的方法。由此，就可通过改造基本抽象模型来建立实时性模型，如具有时序特性的自动机。这就是所设想的时间自动机[2]和其他形式自动机（定时I/O自动机、超时自动机、日历自动机）方式。尽管已经进行了大量的研究工作，尤其是在建立形式化数学基础方面，但在实际应用中的进展却几乎停滞不前，没有超过12个以上的连续变量系统。例如，在航空电子设备系统中，一个基本的3-D防撞协议，至少需要6个整型变量来表示飞机和入侵者的坐标，并用6个整型变量来表示3-D速度矢量，这已经完全超出现有技术的能力。

求解线性算术和未解释函数的可满足性模理论（SMT）已用于有效地验证实时性设计[33]。然而，这种方法仍局限于可表示为线性不等式的具有较少离散状态且时序约束的模型。

在混合动力系统验证领域达到较高的成熟度之前，验证人员必须求助于更传统但相对可扩展的技术，如离散状态*模型校验*。因此，首要任务是允许在行为等效的有限模型中进行无限状态空间分析时找到合适的*抽象技术*。

一旦建立一个合适的抽象，接下来的第2个挑战是*解决状态空间爆炸*的问题。一种可能性是开发一种可将状态空间减少到可控规模的还原技术（如部分降阶、对称性还原或合成方法）。另一种是提高模型校验算法本身，正如5.2.6节中提出的饱和算法[18]。这种模型校验方法最初是作为一种用于生成全局异步/局部同步系统状态空间的非常高效的迭代策略，现已逐步完善以能够解决更多类型的问题。在最新的版本中，可支持快速可变域计算（变量值的上下边界值先验未知），另外，还能分析正如上述所需的不满足“Kronecker一致性”的模型。

接下来，介绍在工业应用领域应用上述方法（即抽象模型的离散状态模型校验）的经验：NASA的飞行跑道安全监控（RSM）软件[35]。该经典方法似乎已在业内保持了一种良好的纪录，这可通过ONERA和空中客车（Airbus）公司的最近报告[5,51]表明。

在此，并不打算将该方法与其他形式化验证工具和技术进行*比较*，因为这不是工业中的习惯做法，其中，工具和人员技能严格确定了每个项目中的验证技术。然而，会尽量详细地证明建模和分析决策，并分享在已成功应用的项目中获得的观念和经验教训。

5.2 飞行跑道安全监控应用

RSM是RIPS [40]中的一部分。通过将由洛克希德·马丁（Lockheed Martin）

公司的工程师设计并实现的 RSM 整合到集成显示系统(IDS)[4]，自 1993 年以来，NASA 就已开发一套驾驶舱应用系统。IDS 还包括其他冲突检测和消解算法，如 TCASⅡ[44]。IDS 的设计使得 RSM 能够充分利用现有的数据通信设备、显示系统、全球定位系统、地面监控系统信息和数据链接。

在实际操作中已采用防撞协议。自 1994 年以来，就一直在使用 TCAS[44]，现在，FAA 要求应用于所有美国商用飞机上。TCAS 具有完整的形式化规范，但由于其复杂性[11,37]，目前只有一部分被验证。其他协议，如 NASA Langley 开发的有助于确保通用航空飞行器在无塔台机场安全着陆的小型飞机运输系统(SATS)[1]，以及一种已经过形式化验证的用于并行着陆场景的替代算法[32]——机载信息横向间距（AILS)[10]。

5.2.1　RSM 的作用

RSM 的目标是检测跑道入侵，这由 FAA 定义为“在机场发生的包括飞机、车辆、人员或地面上物体在内的任何导致产生碰撞危险或使得飞机起飞、准备起飞、着陆或准备着陆时产生分离损失的事件”。

由于航空安全事故大多发生在跑道上或跑道附近，因此 RSM 在避免事故发生方面起着关键作用。RSM 并不是为了*预防*入侵，而是*探测*到入侵行为并*提醒*飞行员。预防功能是由具有某些 IDS 功能的 RIPS 其他组件提供，如头盔显示器、电子运动地图、驾驶舱交通信息显示以及滑行路线。由洛克希德·马丁公司进行的实验研究[35,53]表明，若飞机采用 IDS 技术，则入侵情况不太可能发生。RSM 应能大大提高这种积极作用。

5.2.2　RSM 的设计

RSM 算法的顶层架构如图 5.1 所示。在一个安装在驾驶舱内的设备上运行 RSM 并要求在飞机在机场起飞和降落到机场之前启动该程序。在每架飞机上运行 RSM 的一个独立副本，并将正进行操作的飞机看作*所有者*，而将其他飞机、正在使用同一跑道的地面车辆甚至是像设备这样的物理障碍看作*目标*。

RSM 监控飞机起飞和着陆的跑道周围区域的交通状况。该区域为一个跑道横向边缘达到 220ft、高度为跑道海拔 400ft、长度为每条跑道端 1.1n mile（海里）的三维空间（这对应于飞机起飞和降落轨迹的一个 3°滑行斜坡)。

利用 C 语言编程实现的协议包括重复循环的 3 个阶段。在第 1 阶段中，RSM 采集由数据链路接收到的雷达更新交通信息。可辨识监控区域内的每个目标，并用保存其 3-D 物理坐标。更新频率可能不固定，造成丢失更新信息，由此导致数据可能错误。在该研究中没有解决数据链接错误或遗漏的影响作用，这为今后的研究提出了一项具有挑战性的任务。这些错误已经成为一些测量实验的主要研

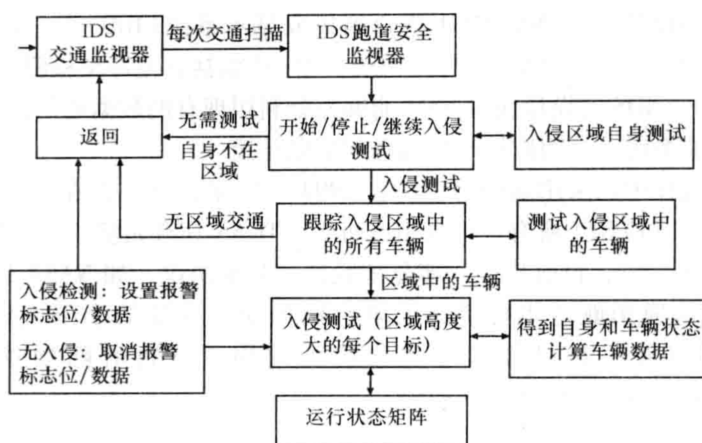


图 5.1 RSM 算法顶层设计

究内容[53]，但这些分析都需要考虑本模型中未关注的随机性，而在本模型中只关注逻辑错误。

在第 2 阶段，利用算法根据预定义的一组值来确定每个目标的状态：滑行、起飞前、起飞、爬升、着陆、弹射和飞行等模式。将在 5.3 节中详细讨论这些状态的具体含义。

第 3 阶段主要是负责入侵检测，并根据自身和目标对象的空间属性（位置、朝向和加速度）以及某些逻辑条件来对每个目标执行。5.3.2 节中的表 5.1 给出了这个阶段的*操作状态矩阵*。安全性分析主要着重于验证表中所列决策准则是否能够检测到所有可能发生的入侵场景。

5.2.3 RSM 的形式化验证

在此，采用基于模型的方法来验证，而不是试图验证实际的 C 代码本身，主要有两个原因：首先，主要安全性能（“无漏警”）的展示需要环境表示（如仅由软件跟踪的目标运动，而在代码中没有表示控制目标的动态规律）；其次，不清楚时序逻辑模型校验能否处理这种规模的代码（与其他已应用于航空电子软件的静态分析技术[28]相比）。

本书认为，对于该应用，一个离散状态模型已足够（以及对于类似的航空交通监控应用，只是具有在途中或机场两种状态）。因为在通过软件可观测的任何行为下，系统状态都是离散的，这是由于飞机位置是每隔固定时间获取的，而不是连续的。

在描述提取的 RSM 模型前，先回顾一下过去几年中所开发的符号验证技术的主要特点。

5.2.4　符号模型校验结构

在对大型复杂系统应用符号模型校验时，内存和时间的需求会很快成为一个巨大障碍。为缓解该问题，可尝试利用所研究系统的特殊性质。称为*结构化符号模型校验*[14,23]的技术在应用于全局异步/局部同步系统时非常成功[12]。其主要思想是*局部*识别（即事实上，大多数事件只影响少数状态变量），并使用该方法来实现饱和式定点迭代，以代替传统的广度优先迭代（即以一个特定的位置引导顺序来搜索事件，可大大降低决策图的峰值大小）。

将离散状态模型表示为一个三元组（$\mathcal{S}_{pot}$，$\mathcal{S}_{init}$，$\mathcal{N}$），其中，$\mathcal{S}_{pot}$为模型的*潜在状态*集合，$\mathcal{S}_{init}\subseteq\mathcal{S}_{pot}$为*初始状态*集合，$\mathcal{N}$：$\mathcal{S}_{pot}\to2^{\mathcal{S}_{pot}}$为在单步下从每个状态可达的指定状态的下一个*状态函数*。在实际应用中，以某种高度形式化方式（如 Petri 网或进程代数）来紧凑地描述该模型，但$\mathcal{S}_{pot}$的规模通常会很大。接下来，一个基本问题是确定从初始状态经重复执行下一个状态函数真正可达的状态集合。从形式化上，相对于$\mathcal{N}$，（可达）状态空间$\mathcal{S}_{rch}\subseteq\mathcal{S}_{pot}$是封闭$\mathcal{S}_{init}$的最小超集，即

$$\mathcal{S}_{rch}=\mathcal{S}_{init}\cup\mathcal{N}(\mathcal{S}_{init})\cup\mathcal{N}(\mathcal{N}(\mathcal{S}_{init}))\cup\cdots=\mathcal{N}*(\mathcal{S}_{init})$$

其中，“ * ”表示自反传递闭包，且 $\mathcal{N}(\chi)=\cup_{i\in\chi}\mathcal{N}(\boldsymbol{i})$。

构建模型的第一步是将模型分解为 K 个*子模型*，或换句话说，即将一个（全局）状态 i 记为 K 元组(i_K，…，i_1)，其中，i_k为子模型 k 的*局部*状态，其中 $K\geqslant k\geqslant1$。因此，潜在状态空间可由 K 个局部状态空间叉乘而给定，即$\mathcal{S}_{pot}=\mathcal{S}_K\times\cdots\times\mathcal{S}_1$。

假设，此时已知每个$\mathcal{S}_k$，可将其中每个局部状态 i_k 映射到范围为 {0, 1，…，n_k-1} 中的序号 i_k，其中 $n_k=|\mathcal{S}_k|$。然后，就可确定$\mathcal{S}_k$，且 k 的取值为{0，1，…，n_k-1}，并对$\mathcal{S}_{pot}$上准降阶多路决策图（MDD）[41,42,48]中的任意集合$\chi\subseteq\mathcal{S}_{pot}$进行编码。

在形式化上，MDD 是一个有向非循环的边标记多重曲线图，其中：

- 每个节点 p 属于 {K，…，1，0} 中的同一层。如果某层为 $K>0$，也可认为该节点是指*变量* x_k；
- 层次 0 只能包含两个*终端*节点 Zero 和 One；
- 如果存在一个无输入弧的唯一*根*节点 r，则该节点位于层次 K，或为终端节点 Zero；
- 一个位于层次 $k>0$ 的节点 p 具有输出边 n_k，标记为 $0\sim n_k-1$。由点 i_k标记的边指向节点 q，该节点为终端节点 Zero，或位于层次 k-1 的一个节点，记为 $p[i_k]=q$；
- 在此没有*重复*节点：如果 p 和 q 均位于层次 k，且对于所有 $i_k\in\mathcal{S}_k$，满足

$p[i_k]=q\ [i_k]$，则 $p=q$。

值得注意的是，MDD 是一种非常成功的 BDD 的泛化[6]，其中，每个节点仅限于二元选择，因此，只要 $n_k>1$，就需要一个多重布尔变量来对单个序号 i_k 进行编码。

MDD 通过以下定义的递归公式来对一组状态$\mathcal{B}(r)$ 进行编码：

$$\mathcal{B}(p)=\begin{cases}\Phi & p=\text{Zero}\\ \{i_1:p\ [i_1]=\text{One}\} & p\text{ 位于层次 }1\\ \bigcup_{i_k\in\mathcal{S}_k}\{i_k\}\times\mathcal{B}\ (p[i_k]) & p\text{ 位于层次 }k>1\end{cases}$$

在符号算法中，多组状态$\chi^{(1)}$，…，$\chi^{(m)}$，即$\mathcal{S}_{\text{pot}}$的所有子集，应同时保存。这是通过 MDD 来*预测*所存储的这些 MDD 的根节点 $r^{(1)}$，…，$r^{(m)}$，并使得这些根节点*共享*来实现，由此可避免重复，即使在不同的 MDD 之间。所定义的一个 MDD 的基本特性是典型性（canonical）：对于一个给定的变量顺序 x_K，…，x_1，表示同一集合的两个节点是同构的，且当采用 MDD 预测时，这种情况可利用哈希表（hash table）技术在 O(1) 时间内很容易地确认。

除了紧凑表示 MDD 所提供的状态集合，符号方法还需对下一状态函数进行紧凑表示。由于$\mathcal{N}$可看做是一组节点对(i,j)，其中$j\in\mathcal{N}(i)$，由此可利用一个包括所有“起始”变量 x_K，…，x_1和“终止”变量 y_K，…，y_1的层次为 $2K$ 的 MDD 来保存。然而，这种单片 MDD 编码通常需要大量内存。接下来，构建模型的第 2 步是将$\mathcal{N}$分解为下一状态函数的*析取*[9]：$\mathcal{N}(i)=\bigcup_{\alpha\in\varepsilon}\mathcal{N}_\alpha(i)$，其中 ε 为*事件*的有限状态集，$\mathcal{N}_\alpha$是与事件 α 相关的下一状态函数。此时，可认为当事件 α 在状态 i 发生或*激发*时，$\mathcal{N}_\alpha(i)$ 为一组可进入的系统状态，且如果$\mathcal{N}_\alpha(i)=\Phi$，则 α 在状态 i 无效，反之，则使能。通常，在具有变量交叉排序（x_K，y_K，…，x_1，y_1）的 MDD 森林中保存$|\varepsilon|$MDD 可大大改善保存$\mathcal{N}$[20]所需的内存要求。

这种结构化方法根据 K 个状态变量进一步执行，并对每个$\mathcal{N}_\alpha$ 划分。一种有效方法[17,48]是采用由马尔可夫链（Markov chains）[7]启发的 Kronecker 表示法[31]，假设模型为 *Kronecker 一致*，即$\mathcal{N}_\alpha$可*共同*分解为 K 个局部下一状态函数$\mathcal{N}_{k,\alpha}$：$\mathcal{S}_k\to2^{\mathcal{S}_k}$，其中，$K\geqslant k\geqslant1$，且满足：

$$\forall(i_K,\cdots,i_1)\in\mathcal{S}_{\text{pot}},\mathcal{N}_\alpha(i_K,\cdots,i_1)=\mathcal{N}_{K,\alpha}(i_K)\times\cdots\times\mathcal{N}_{1,\alpha}(i_1)$$

定义 $K\cdot|\varepsilon|$矩阵为 $\boldsymbol{N}_{k,\alpha}\in\{0,1\}^{n_k\times n_k}$，且 $\boldsymbol{N}_{k,\alpha}\ [i_k,i_k]=1\Leftrightarrow j_k\in\mathcal{N}_{k,\alpha}(i_k)$，将$\mathcal{N}_\alpha$编译为一个 Kronecker 积：$j\in\mathcal{N}_\alpha(i)\ \Leftrightarrow\bigotimes_{K\geqslant k\geqslant1}\boldsymbol{N}_{k,\alpha}\ [i_k,i_k]=1$，其中，状态 i 可作为$\mathcal{S}_{\text{pot}}$中的一个*混合基序号*，且$\otimes$表示矩阵的（布尔型）Kronecker 积。

$\mathcal{N}$的 Kronecker 编码在内存方面也是相当有效的，这是由于其可用于确定和搜索*局部事件*。若 $\boldsymbol{N}_{k,\alpha}=\boldsymbol{I}$（单位矩阵），则认为事件 α 是与层次 k 无关的。对于

全局异步/局部同步模型，大多数事件仅影响少量状态变量而与其他状态变量无关。只有矩阵 $\boldsymbol{N}_{k,\alpha} \neq \boldsymbol{I}$ 需要真正存储，在实际应用中，其存储量通常为 $O(|\varepsilon|)$ 而非 $O(K \cdot |\varepsilon|)$。此外，这些矩阵在实际中是极其稀疏的，并且通常仅需要 $O(n_k)$ 量级的内存而非 $O(n_k^2)$。

5.2.5　符号状态空间生成饱和算法

确定局部性事件不仅对于$\mathcal{N}$的 Kronecker 表示至关重要，同样，对于*饱和*算法也是如此[18]。设 $\text{Top}(\alpha)$ 和 $\text{Bot}(\alpha)$ 分别表示 $\boldsymbol{N}_{k,\alpha} \neq \boldsymbol{I}$ 时的最高层和最低层。如果对于使得 $\text{Top}(\alpha) \leqslant k$ 的所有事件 α，为一个定点，则可认为位于层次 k 的一个节点 p 是*饱和*的，即

$$\forall i_k, \cdots, i_{k+1} \in \mathcal{S}_K \times \cdots \times \mathcal{S}_{K+1}, \text{Top}(\alpha) \leqslant k \Rightarrow$$
$$\{(i_K, \cdots, i_{k+1})\} \times \mathcal{B}(p) \supseteq \mathcal{N}_\alpha(\{(i_K, \cdots, i_{k+1})\} \times \mathcal{B}(p))$$

这可通过定义$\mathcal{N}_{\leqslant k} = \cup_{\alpha:\text{Top}(\alpha) \leqslant k} \mathcal{N}_\alpha$，并将$\mathcal{N}_\alpha$应用于一组 $l \geqslant \text{Top}(\alpha)$ 的$\chi \subseteq \mathcal{S}_l \times \cdots \times \mathcal{S}_1$中，可简写为$\mathcal{B}(p) \supseteq \mathcal{N}_{\leqslant k}(\mathcal{B}(p))$，这是因为$\mathcal{N}_\alpha$保持所有大于 $\text{Top}(\alpha)$ 的局部状态不变。

饱和度算法首先采用 MDD 对初始状态$\mathcal{S}_{\text{init}}$编码，并自下而上使得节点饱和。一个简化的算法高层伪代码如图 5.2 所示，其中，$\varepsilon_k = \{\alpha: \text{Top}(\alpha) = k\}$。

采用多重嵌套的简单定点迭代来实现饱和，不同于传统的广度优先方法，即在每次迭代过程中，将每个$\mathcal{N}_\alpha$应用于整个已知状态集合，或应用于整个新出现状态集合。只要对位于层次 k 的某个节点 p 执行饱和操作，则其子节点均已饱和。为使得节点 p 饱和，该算法对满足 $\text{Top}(\alpha) = k$ 的事件 α 中所有节点 p 重复激活，直到达到某一定点。且满足附加条件：如果在层次 k 之下激活创建新的节点，则在这些节点实现节点 p 饱和之前立即饱和。

文献 [18] 的结果表明在内存和时间两个方面，该方法均比广度优先的符号搜索方法提高几个数量级，使得这成为在全局异步/局部同步的离散事件系统中状态空间生成的最有效的符号定点迭代策略。

尽管所提算法假设在调用*状态空间生成*之前已知局部状态空间$\mathcal{S}_k$，但这并非一个必要条件。现已对饱和算法进行扩展，使得可通过所提出的局部符号状态空间生成交叉结合显式局部状态空间搜索来实时发现新的局部状态[19]。在这种情况下，算法实现基本上更加困难，但理论复杂性保持不变，且随着建模人员不再考虑局部状态变量的边界，在实际应用中大大提高建模容易程度。

```
StateSpageGeneration (r)                       r为MDD编码S_init的根节点
1.for k=1 to K do                              自下而上使得初始节点饱和
2.对于每个MDD根节点为r的位于层次k上的节点p执行Saturate(p)
```

```
Saturate (p)                                   设k为节点p所在层次
1.重复执行                                     更新p，使之满足固定点
   B(p)=B(p)∪N≤k(B(p))
2.选择一个i_k∈S_k，使得满足p[i_k]≠Zero，以及一个α∈ε_k使得满足N_α(i_k)≠Φ
{i_k}×B(p[i_k])中的一个状态可使能α
3.对于每个 j_k∈N_α(i_k)，设置p[j_k]到Union(p[j_k],Firesat(α,p[i_k]))
 更新p位于正确位置
4.直到p不再变化
5.返回MakeUnique(p)      如果p多次重复一个节点d,则去除p并返回d
```

```
FireSat (α,p)                 设k为节点p所位于的层次，注意Top(α)>k
1. 如果Bot(α)>k，则返回p                       递归激发的终止条件
2. 如果已预先计算f=FireSat(α,p)，则返回f
查找FiringCache
3. 创建一个位于k层次的新节点f，且所有边指向Zero
 不能调节p位于正确位置
4. 对于每个i_k∈S_k，使得满足p[i_k]≠Zero，且对每个j_k∈N_α(i_k)执行
5. 设置f[j_k]到Union(f[j_k]),Firesat(α,p[i_k]))
…但可以调节节点f到正确位置
6. 返回Saturate(f)
```

```
Union(p,q)                                     设k为节点p和q所位于的层次
1. 如果p=Zero或q=One或p=q，则返回q
2. 如果q=Zero或p=One，则返回p
3. 如果已预先计算u=Union(p,q)，则返回u
查找UnionCache
4. 创建一个位于层次k的新节点u，且所有边指向Zero
5. 对于每个i_k∈S_k设置u[i_k]到Union(p[i_k],q[i_k])
6. 返回MakeUnique(u)   如果u多次重复一个节点d，则去除u并返回d
```

图 5.2 利用 MDD、Kronecker 和饱和迭代来生成状态空间

5.2.6　基于饱和算法的模型校验

采用下一状态函数的 Kronecker 表示的优点还可应用于状态空间生成的广度优先迭代，以及应用于符号计算树逻辑（CTL）模型校验的一般迭代[25,46]，后者更具挑战性。

众所周知，一个 CTL 公式可用 EX、EU 和 EG 3 个运算符，以及布尔操作符 ¬、∧和∨来表示。显然，饱和算法不适用于 EX 运算符的计算，这是由于这会涉及下一状态函数$\mathcal{N}$的每一步应用，而包含饱和算法的任何定点算法关注于在模型进化步骤中的任意长序列。类似地，EG 运算符从一组满足特定条件 p 的状态$\mathcal{P}$开始，从$\mathcal{P}$中迭代去除迁移到状态$j\in\mathcal{P}$的任一状态 i。也就是说，不同于状态空间生成，其中一个状态 j 可添加到一组状态集$\mathcal{S}_{\text{rch}}$中，如果存在任一对于某些状态 $i\in\mathcal{S}_{\text{rch}}$满足$j\in\mathcal{N}_{\alpha}(i)$ 的事件 α，在 EG 迭代需求中移除某个状态的条件是需要校验*所有*事件。因此，饱和算法不可能直接用于 EG。

然而对于 EU 运算符，情况好很多[23]。首先，由于 EFq 等价于 $E[\text{true}Uq]$，因此，可将 EF 运算符看作 EU 的一个特殊情况。为符号计算一组满足 EFq 条件的状态，可从满足状态条件 q 的集合$\mathcal{Q}$开始，“赶着模型反向走”，并对$\mathcal{Q}$增加可迁移到$\mathcal{Q}$中某一状态的所有状态。这正是状态空间生成问题，其中，$\mathcal{Q}$为初始状态集合，且下一状态函数为模型原始下一状态函数的逆，即 $i\in\mathcal{N}^{-1}(j)$，当且仅当$j\in\mathcal{N}(i)$，因此饱和算法可自然而然地利用其所有优点。值得注意的是，通过$\mathcal{N}$的 Kronecker 编码，足以利用转置矩阵 $\boldsymbol{N}_{k,\alpha}^{\text{T}}$来描述模型的反向进化。

然而，在处理计算满足条件 $E[p\cup q]$ 的状态的一般问题时，不能直接应用原始的饱和算法，这是因为必须保证从$\mathcal{Q}$中状态反向进入时，不会偏离$\mathcal{P}$中的状态（即满足 p）。文献［23］提出的解决方案是将事件 ε 分为安全事件 ε_{S}和不安全事件 ε_{U}。前者集合的显著特征是对于任何安全事件 α 和任何状态 $j\in\mathcal{P}\cup\mathcal{Q}$，$i\in\mathcal{N}_{\alpha}^{-1}(j)$ 同样意味着 $i\in\mathcal{P}\cup\mathcal{Q}$，即通过激发 α 不可能从一个不满足 p 或 q 的状态运动到一个满足 p 或 q 的状态。对所有事件进行分类的代价，本质上就是一种广度优先迭代（每个事件必须反向激发一次）。然后，可通过用一种后向广度优先步骤（即具有 ε_{U}事件集合的 EX）来代替对于安全事件的反向饱和步骤（即具有 ε_{S}事件集合的 EF）来计算满足 $E[p\cup q]$ 的状态集合，直到不再增加状态。这种基于饱和的 EU 算法的效率取决于模型，尤其是取决于有多少事件归类为安全事件。在最佳情况下，所有事件都是安全的（如在 EF 情况下），而在最坏情况下，没有事件是安全的（复杂性就是传统的符号广度优先算法，尽管仍具有$\mathcal{N}$由 Kronecker 表示的优势）。

最近，提出了一种用于饱和算法进行 EU 计算的另一种方法[55]，其中，通过与$\mathcal{P}\times\mathcal{S}_{\text{pot}}$的相互作用，在进行反向时通过保持约束到$\mathcal{P}$中来调节每个下一状态

函数$\mathcal{N}_{\alpha}$，在开始计算前或在定点迭代中实时动态执行。该新方法比文献［23］中的方法更加有效，但它没有被发现的时候，进行了在本章中所描述的研究。

5.2.7 随机模型校验可靠性和定时分析工具（SMART）

这里所介绍的符号技术在 SMART[16] 工具中得以应用。给定将一个系统描述为扩展 Petri 网[49] 情况下，SMART 可生成状态空间，验证时序逻辑特性，并计算定时分析和随机分析的数值解。在 SMART 中实现了大量的显式方法和符号方法。除了利用 MDD 来对状态集合进行编码，并通过 Kronecker 运算符、矩阵图[21,47] 或 MDD 来对下一状态函数进行编码之外，SMART 还利用一种特殊形式的边缘值 MDD（EV^{+} MDD）[22] 来生成 CTL 模型校验中的最小长度反例和证据。

SMART 输入是一个具有图灵等价扩展（直接迁移、标志相关弧基数和迁移保护）[49] 的有限状态空间的扩展 Petri 网。每个 SMART 输入文件都定义了一个或多个结构化（即划分为子模型）模型。一个模型可参数化并定义一系列在本例中认为是通过搜索系统状态而评价为逻辑查询的测量值。

可在网上获取 SMART 的用户手册[15]。

5.3 RSM 的离散模型

5.3.1 整型变量和实型变量抽象化

自动抽象技术是非常期望的且已成功地用于许多应用领域。最著名的相关技术包括抽象解释[29]、域抽象[30]、数据抽象[26,27,38]、谓词抽象[34,45]、自动抽象细化（CEGAR）[8,24]、近似处理[3,50,52]、数据等效[13]、切片[36] 和常规无限状态空间校验[54]。

由于手动抽象通常需要完全理解系统及其发展变化，因此一直以来视为非常棘手的问题。然而，在许多情况下，可根据所用工具的性能特点来定制手动建立的模型。因此，如果建模人员对整个系统过程具有良好的*控制*，那么手动抽象通常也能产生较好的结果。但随着自动化的高度需求，非常普通的自动抽象方法都会导致手动抽象逐渐消失。例如，所生成的抽象可能效率较低、不准确，或在 CEGAR 情况下易受芝诺（Zeno）效应的影响。

此外，在本例的特定情况下，需要根据全局异步/局部同步系统的分析来确定饱和目标。通过连续化环境的建模以及在模型中具有高度局部事件特性的离散系统事件，可利用基于饱和的模型校验方法成功地构建抽象模型的状态空间。尽管不再需要降阶技术（部分降阶、对称、谓词抽象），但其中的某些技

术可能与离散化方法完全正交，从而可用于降阶模型的顶层。同时，离散化程度并不那么剧烈（正如在应用谓词抽象的情况下），系统行为仍然非常接近于实际的航天器轨迹。

5.3.2　RSM的SMART模型

与采用可最终将整个三维空间分成（可能不规则，但逻辑等价）很多区域的谓词抽象技术不同，在此确定固定个数的变量（3个坐标轴上的坐标），然后将这些变量的域分成若干段。在进行上述操作中，需考虑两个因素：首先，在驱动谓词抽象的协议中，没有对所需设定的特性集进行任何实际的形式化规范；第二，离散化类型可产生与空间中航天飞机运动的几何结构非常接近的模型执行结果，从而可使得没有接受过培训或可能不了解抽象技术的设计工程师更好地理解。

最后，通过将模型划分为$n+1$个子模型来选择系统变量，其中n为区域内移动目标的个数。子模型0的变量描述其自身状态，而其余n个子模型描述每个目标的状态。对于子模型i，$0\leqslant i\leqslant n$，具有以下相关属性：

位置：一个三维矢量（x_i，y_i，z_i），其中，X轴为跑道宽度方向，Y轴是沿跑道长度方向，Z轴为与跑道垂直的方向。

速度和航向：另一个三维矢量（vx_i，vy_i，vz_i）。

沿跑道方向的加速度：ay_i。

姿态：一个枚举型变量，$status_i$。

报警标志位：一个布尔型变量，$alarm_i$。

阶段：一个枚举型变量，$phase_i$。

相应的变量域如下（为阅读方便省略下角i）：

- 坐标x，y，z可简单取为$x, y, z\in\{0, 1, 2\}$，其中0表示监控区之外，1表示在监控区附近，2表示在跑道区域。然而，在此选择一种更精细的表示：$x\in\{0, \cdots, max_x\}$，$y\in\{0, \cdots, max_y\}$，$z\in\{0, \cdots, max_z\}$，其中，0意味着在监控区之外，而常量max_x，max_y，max_z可根据建模人员的偏好进行调整。也就是说，位置（0，0，0）表示在区域之外的所有位置。这是一个退出该区域或根本没进入该区域的目标的位置。
- 速度值vx，vy，vz可赋值在域$\{0, \pm1, \pm2\}$内，其中0表示无运动，±1意味着缓慢运动（低于预定的滑行速度阈值TS，45节），±2意味着快速运动（大于阈值TS）。同样，一种采用另一个参数max_{speed}的更精细表示为$vx, vy, vz\in\{-max_{speed}, \cdots, 0, \cdots, max_{speed}\}$。

- 加速度 a_y 只有两个相关值：非负和绝对负。
- 姿态包括 {场外，滑行，起飞，爬升，降落，弹射，飞行}。
- *阶段*包括 {雷达更新，姿态设置，检测}。

变量*阶段*作为 3 个循环步骤中每个步骤所执行算法的一个程序计数器：

阶段 = 雷达更新：更新目标位置。

阶段 = 姿态设置：更新目标姿态。

阶段 = 检测：设置或复位报警标志位。

接下来，讨论上述 3 个步骤中各个步骤的建模决策。

5.3.2.1 三维运动目标

所提出的离散化方法将监控空间划分为一个三维网格。航天器可能处于的位置是有限个数的网格单元，离散域 $\{(0,0,0)\} \cup \{1,\cdots,max_x\} \times \{1,\cdots,max_y\} \times \{1,\cdots,max_z\}$。同理，连续轨迹也必须由三维网格的离散轨迹表示。在此考虑 3 种方案。

首先是投影法，即将每条可能的连续轨迹对应于网格中相应的离散路径。一个投影法示例如图 5.3（为便于可读性起见，在二维空间中表示）所示。图中具有 100 个单位大小（ft）的网格单元，每 0.5 个时间单位（s）采集一次。速度单位是在每次更新遍历后对坐标轴分割测定的。这种方法存在的问题是难以区别物理的可能性和不可能性。在此，没有一种有效方式来排除所有异常。例如，一个目标可以改变其实际位置，而离散位置却未改变。同时，也难以确定一个目标位于某一网格中的速度和单位时间个数之间的相互关系：可能是一次运动（高速）或多次运动（低速），但在离散模型中无法计算一个网格内所允许的单位时间个数的上限。

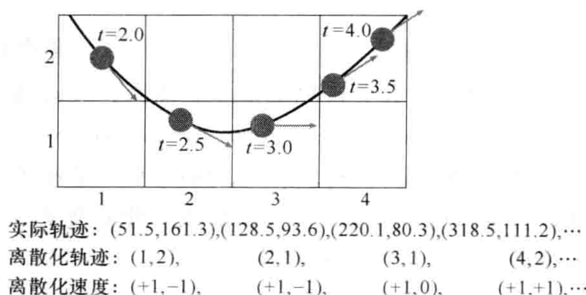


图 5.3 二维空间中连续轨迹的投影示例

因此，在此考虑用一种已证明更具有实用性的不同办法来对运动目标建模。一种办法是允许目标几乎是自由运动，即在某种意义上总是允许运动到相

邻网格。原则上，一个目标可留在当前网格内或以在坐标 x，y，z 中不确定减少、不变、增加来运动到相邻 26 个网格中的任何网格中。然而，这种变化必须与航向保持一致。另一方面，应限制只能允许在不包含大量轨迹（在物理上大多数是不可能的）的相邻网格之间迁移。同时，认为模型中没有不包含实际轨迹。如果网格单元足够大，则上述成立。在这个包含所有特定属性的最简单模型中，网格单元的大小为 900ft。给定每 0.5s 在数据链路上位置更新一次，则只有当目标运行速度大于 1800ft/s，即约 1227mile/h（或约 1975km/h）时，目标才能跨越一个网格单元，并运动到远离两个离散位置的一个网格单元。此时的速度超过音速的 1.6 倍。尽管假设在民用机场跑道上不存在这种速度并非完全可靠，但从模型中排除这种情况是非常合理的，同时还有助于简化分析。另外，由于可能存在的状态个数变得可控，粗略的离散化还能缓解状态空间爆炸的问题。图 5.4 表明在第 2 种模型中（为简便起见，也以二维空间表示）目标可能存在的运动。

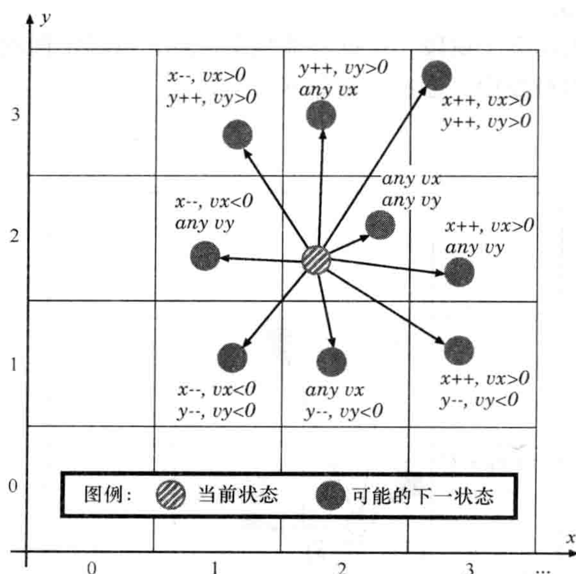


图 5.4　一个目标在二维空间中可能存在的运动（“自由运动”模型）

第 2 种模型可能仍包含有不现实的轨迹，如在两个相邻网格单元间来回摆动（相应的速度分量在正向和反向之间来回交替变化），或甚至在正向速度下一直停留在同一个网格单元中。

如果需要完全彻底地消除不必要的轨迹，应考虑禁止速度突然变化的第 3 种

方案。也就是说，坐标 x，y，z 和速度分量 vx，vy，vz 至多只能有一个在绝对值上变化。进一步可限制在每个时间点处只能允许速度增大、减小和不变，且坐标相应更新：如当速度分量 vx 非负时，变量 x 不能减小。

与自由运动模型相比，图 5.5 给出当前状态下，速度为 $vx=3$ 和 $vy=3$ 的一个目标可能处于的下一状态（二维空间中）。在这种情况下，对应于 x 和 y 不变或相互独立的增大，只有 4 个可能的新位置。所减少的选择个数是由于速度只能为正，而不允许在负轴方向上的任何运动。只有在当前状态下速度分量为 0 时，目标才能在相应坐标轴的方向上运动。从两个随后的观测位置可推导出速度，从而目标需从一个网格单元运动到相邻的另一个网格单元（例如，从 $x=3$ 到 $x=4$），即使在这个方向上的相应速度为 0（$vx=0$）；在下一步中，速度上将会体现这一变化（$vx=1$），如图 5.6 所示。在这个模型中，至少需要两个步骤使得速度从正向变为负向（反之亦然）。这意味着不可能出现“之字形运动”，这在分析中具有非常重要的作用，正如在 5.3.3 节中所述。

5.3.2.2 姿态定义

在循环执行的第 2 个阶段，根据其他状态信息，可确定性地更新每个航天器的姿态变量。在本模型中，姿态变量如下：

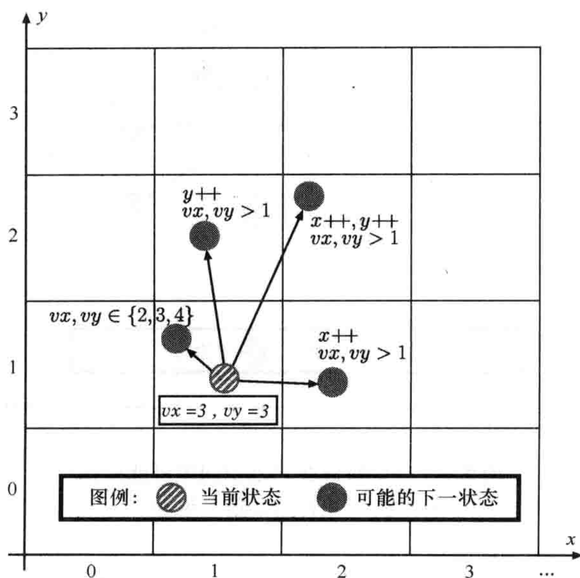


图 5.5 满足 $vx=3$ 和 $vy=3$ 时状态可能发生的运动（“限制”模型）

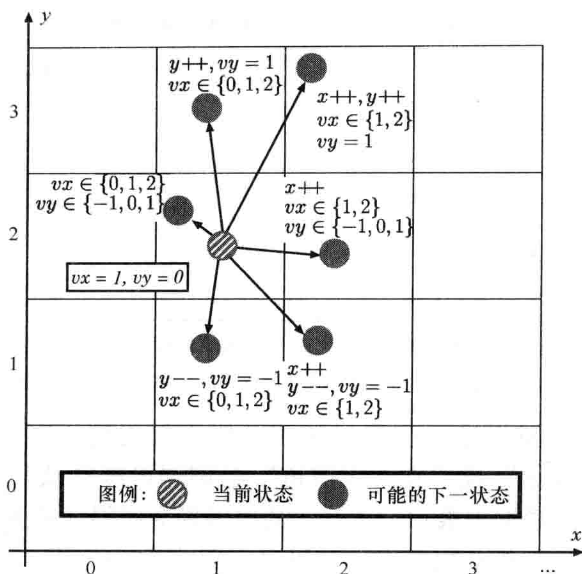


图5.6 满足 $vx=1$ 和 $vy=0$ 时状态可能发生的运动（“限制”模型）

场外：不在监控区内

$\equiv (x=0) \wedge (y=0) \wedge (z=0)$

滑行：在地面上低速或不与跑道航向同向的运动

$\equiv (z=1) \wedge (|vx| \leqslant \mathrm{TS} \wedge |vy| \leqslant \mathrm{TS})) \vee (vx \neq 0)$

起飞：在地面上，与跑道航向同向的加速运动

$\equiv (z=1) \wedge (|vy| > \mathrm{TS}) \wedge (vx=0) \wedge (a_y \geqslant 0)$

弹射：在地面上，与跑道航向同向的减速运动

$\equiv (z=1) \wedge (|vy| > \mathrm{TS}) \wedge (vx=0) \wedge (a_y < 0)$

爬升：在空中，与跑道航向同向以绝对正向垂直速度的运动

$\equiv (z>1) \wedge (vx=0) \wedge (vz>0)$

降落：在空中，与跑道航向同向以负向垂直速度的运动

$\equiv (z>1) \wedge (vx=0) \wedge (vz \leqslant 0)$

飞行：在空中，不是爬升或降落模式

$\equiv (z>1) \wedge (vx \neq 0)$

5.3.2.3 报警标志位设置

RSM 算法的第3个也是最重要的阶段是对每个目标设置报警标志位。在伪代码中，这相当于一个单变量的赋值语句：根据表5.1所示的操作状态矩阵中其他变量当前值的不同组合来设置每个 alarm_i 的值（布尔型）。

表 5.1 RSM 报警标志位设置时的操作状态矩阵

目标→滑行		起飞	爬升	降落	弹射	飞行
自身↓ 滑行	—	a∧f	a∧f	a∧f	a∧c∧f	—
起飞	a∧f	d∨e	d∨e	d∨e	a∨d	b∧c
爬升	a∧f	d∨e	d∨e	d∨e	d∨e	b∧c
降落	a∧f	d∨e	d∨e	d∨e	a∨d	b∧c
弹射	a∧c∧f	a∨d	a∨d	a∨d	d∨e	b∧c
飞行	—	b∧c	b∧c	b∧c	b∧c	—

注：a：接近距离；b：在起飞或降落航道时；c：小于最小间隔的距离；d：在同一方向上低于最低间隔的起飞或降落；e：在相反方向上接近距离的起飞或降落；f：在或接近跑道上滑行或静止。

在 Petri 网中很难对这一相当复杂的赋值语句建模，这是因为两个因素。首先，如“距离接近”或“在起飞航道”的谓词可能涉及几何方程和线性方程，且难以在离散模型中表示。然而，某些因素会使得任务更加容易：设计人员只要坚持简单的解析几何概念，就可避免逐个案例分析。例如，“与目标 i 的距离接近”通常应通过比较当前状态和之前状态的表达式值 $\sqrt{(x_0-x_i)^2+(y_0-y_i)+(z_0-z_i)^2}$ 来进行评估。这进一步意味着每个目标之前的位置应存储在一组辅助变量中，记为 $oldx_i$，$oldy_i$，$oldz_i$，从而进一步扩大了状态空间。然而，这可通过利用由每个航天器获得的姿态信息来避免。例如，如果航天器自身正在滑行，而目标 i 在起飞，可知 $z_0=1$，vy_0，$vy_0 \leqslant TS$，$z_i=1$，$vx_i=0$，$|vy_i|>\mathrm{TS}$，并且 $vz_0=vz_i=0$。即目标在地面上，与跑道对齐，且运动速度超过最高滑行速度。随着距离不断接近，根据航天器自身的运动方法，足以位于目标之前。因此，在这种情况下，谓词可表示为

$$a \equiv (vy_i>0 \wedge y_0>y_i) \vee (vy_i<0 \wedge y_0<y_i)$$

其他谓词可类似地表示如下：

$$\begin{aligned}
b \wedge c \equiv & (vy_0>0 \wedge y_0 \leqslant y_i \leqslant y_0+1 \wedge |x_0-x_i| \leqslant 1 \wedge z_i \leqslant 2) \\
& (vy_0<0 \wedge y_0-1 \leqslant y_i \leqslant y_0 \wedge |x_0-x_i| \leqslant 1 \wedge z_i \leqslant 2) \\
d \equiv & vy_0 \cdot vy_i>0 \wedge |x_0-x_i| \leqslant 1 \wedge |y_0-y_i| \leqslant 1 \\
e \equiv & (vy_0>0 \wedge vy_i<0 \wedge y_i \geqslant y_0) \vee (vy_0<0 \wedge vy_i>0 \wedge y_i \leqslant y_0) \\
f \equiv & 1<x_i<\max_x
\end{aligned}$$

其中，上述示例表达式的推导是针对以下一对状态：$b \wedge c$ 表示起飞——飞行；d 和 e 表示起飞——起飞。

表 5.2 给出了某个目标在基于 SMART 模型下的状态空间测量。表中的缺失项对应于需要过多的运行时或内存的参数选择。

尝试采用其他工具均以失败告终：符号模型校验器 NuSMV 甚至在开始生成之前就内存不足，这是由于迁移关系的二元决策图（BDD）编码太大，而显式模型校验器 SPIN 只能探索可揭示某些问题的非常小的部分状态空间（即使采用部分降阶也小于 $1/10^6$）。

表 5.2 RSM 模型的状态空间生成测量

max_{speed} →	2	3	4	5
↓网格大小	状态空间生成时间（s）			
3×5×4	75.92	105.17	179.28	252.25
3×7×4	195.54	324.65	604.23	805.95
3×10×5	995.18	2212.24	4668.55	7348.27
5×10×7	48257.30	—	—	—
	内存占用（MB）			
3×5×4	11.19	21.20	32.58	49.39
3×7×4	18.27	36.02	56.91	87.25
3×10×5	42.59	83.53	138.56	218.85
5×10×7	246.22	—	—	—

5.3.3 RSM 模型校验

验证工作主要集中于确定表 5.1 中的矩阵操作是否可以保证不存在漏报警情况。由于在协议规范中不包括该必要条件的形式化描述，因此必须探索不同的方式来表示这一特性。

利用如下谓词来定义关键概念（下角 o 和 t 分别是指航天器自身和目标）：

$$\text{detect} \equiv \text{phase}_o = \text{detect} \wedge \text{phase}_t = \text{detect}$$

$$\text{sep} \equiv \text{distance}(o,\ t) > \text{minimum separation}$$

$$\text{alarm} \equiv \text{alarm}_t = \text{true}$$

$$\text{track} \equiv \text{status}_o \notin \{\text{taxi},\ \text{flythru}\} \vee \text{status}_t \notin \{\text{taxi},\ \text{flythru}\}$$

5.3.3.1 安全性能

是否存在不满足最小间隔且报警关闭的跟踪状态？

- 在 CTL 语句中：EF（$\text{detect} \wedge \text{track} \wedge \neg\ \text{alarm}$）

模型校验器返回一个在当前状态下不满足谓词“距离正在接近”的监测状态。这是图 5.7 所示的情况。然而，由于该报警标志可能是在之前状态下设置的，当最小间隔信息第一次丢失时，这可能并不对应于一个不必要行为。报警变量的值取决于报警是否“过时”或不适用于更多的循环周期。

随着在一个特定时间内检测该性能，而不考虑导致当前状态的事件发生序列，可观察到“无记忆”的特性会影响查询结果。为更好地理解这个系统，接下来探讨两个航天器之间违反最小间隔距离后的系统状态。

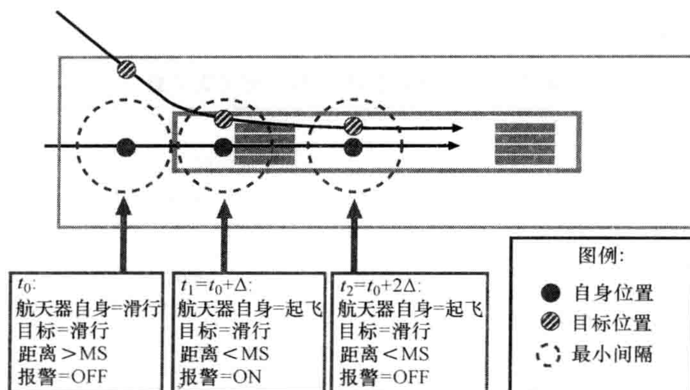


图 5.7 违反第一安全性能的情况（在地面上）

5.3.3.2 造成间隔损耗的状态转移

是否存在向当前状态迁移且报警关闭而导致最小间隔损耗的状态？

- $EF(detect \wedge track \wedge sep \wedge E[(\neg detect) \cup (detect \wedge track \wedge \neg sep \wedge \neg alarm)])$

对于该查询，观察到的是航天器自身在降落或爬升状态，而目标比航天器自身速度更快地以一定角度在间隔距离内运动以从跑道一侧飞越（见图 5.8）。在这种情况下，设置报警的条件是“距离小于最小间隔并且目标位于起飞或降落的轨道上。”在此，第 2 项不满足，所以没有报警。若两者都未接收到报警，那么航天器就可能真正相撞（见图 5.8 中的相交轨迹）。

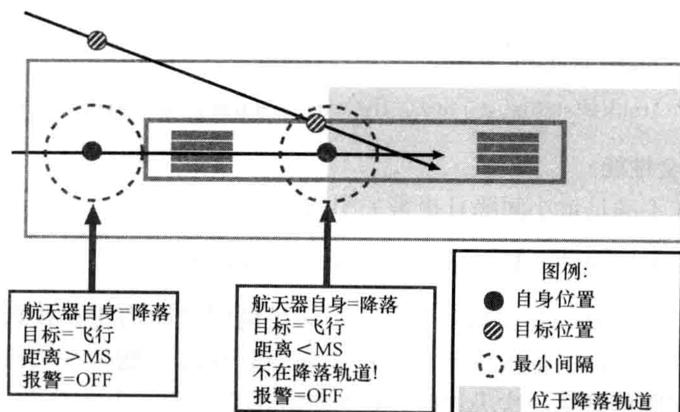


图 5.8 场景 2，空中：飞行目标与降落航天器冲突

这种情况可由 RSM 开发人员通过增加“距离小于最小间隔”作为这一组合状态的部分判断准则来进行修正。

值得注意的是，在所有状态（迁移前和迁移后）中都包含谓词 track，因为入侵定义为至少一个航天器起飞和降落。然而，该附加约束条件可能会屏蔽某些其他不必要行为。因此，接下来探讨一个更一般的性能。

5.3.3.3　更强的安全性能

是否在之前未设置报警情况下可达到一个最小间隔丢失的跟踪状态？

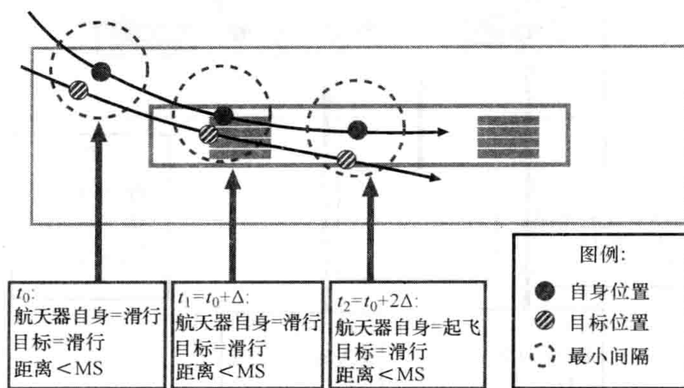


图 5.9　场景 3（在地面上）：滑行目标干扰航天器起飞

- $E[(\neg \text{alarm}) U(\text{detect} \wedge \text{track} \wedge \neg \text{Sep} \wedge \neg \text{alarm})]$

模型校验器得到几种满足上述查询的场景。如图 5.9 所示，航天器可进入监控的滑行区域（与跑道未对齐）且彼此已经非常接近。值得注意的是，算法明确忽略这种状态组合，因为不符合入侵定义。然而，一旦在跑道上，航天器即可改变方向并对准跑道。此后，可归类为起飞阶段（如果在空中，即为爬升）。其余航天器与之保持最小间隔，但未接近：可以位于航天器自身之后，或更危险地在其之前。由于同样不满足准则“距离接近”，因此不会报警。如果在入口处两个航天器之间的距离非常小，就可能没有足够时间执行避让操作，即使随后因过于接近而触发报警。

图 5.9 给出了一个关于最后安全性能的反例。存在一个不跟踪空中状态的完全相同的场景（飞行姿态）。

为确定是否真正关注这种情况，必须在达到可能最坏的状态之后观察可能延续的场景。在下一状态，如果距离不断接近，则会发出警告，同时不再存在“漏报警”的情况。一个恶意智能体使得该问题继续维持的一种方式如图 5.10 所示。如果目标“之字形运动”且每次雷达更新时与航天器自身之间都具有相同的离散化距离，那么目标就可在较长时间内保持在最小间隔半径中。目标必须

之字形运动来保持这一距离，这是因为若与航天器本身沿平行轨道运动将会导致 RSM 认为目标处于起飞状态。对于新组合的操作状态，报警标准是“在同一方向起飞且距离小于最小间隔”。因此，只要目标停止之字形运动，就会马上报警。

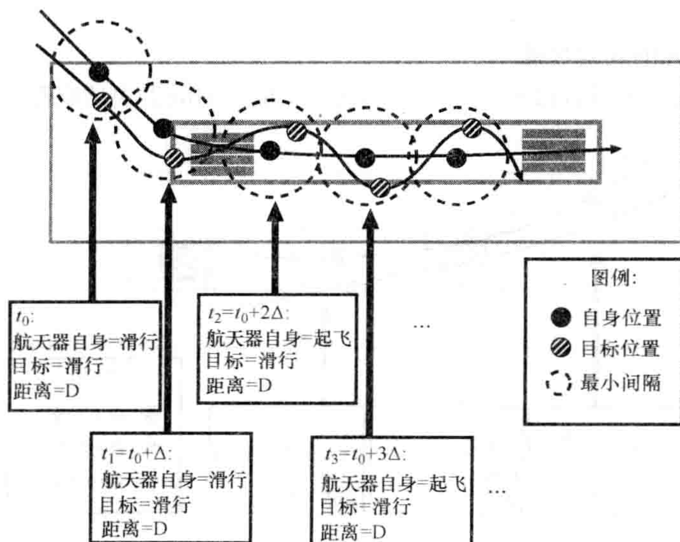


图 5.10　场景 4（在地面上）：航天器自身起飞时目标在其周围之字形运动

当目标不是航天器而是如服务卡车等车辆时，会对目标的恶意行为增加自由度（见图 5.11 中的场景 5）。最初，通常认为地面车辆是在按照协议处于滑行模式，而无需考虑其速度、航向和物理坐标。因此，正如在场景 4 中，目标能够近距离跟随航天器本身，甚至在航天器本身在准备起飞和加速时仍能继续跟踪。

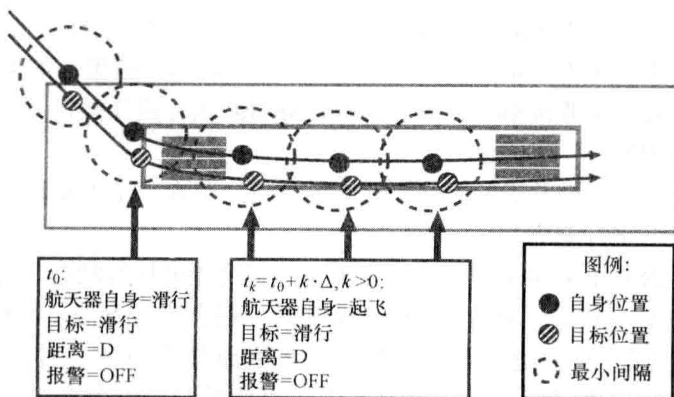


图 5.11　场景 5：地面车辆尾随航天器自身起飞的阴影

与场景 4 中的相同原因，同样不会产生报警标志。RSM 开发人员考虑到该问题，并消除了对地面车辆的特殊处理。这在场景 5 中进行了详细阐述。

场景 4 中的情况并未解决。这是因为在实际中很难实现，即使是一个非常熟练的破坏者故意这么去做，因此这种情况极少关注。与此同时，也存在着一些有益的地方：设计师意识到这是小概率事件。另外，这是系统中唯一剩下的不必要行为，可用于验证 RSM 算法的阶段 3。

5.4 探讨

5.4.1 经验教训

尽管具有不可避免的缺点，但 RSM 的检验具有不可否认的价值。RSM 算法的设计人员已列出一系列在测试过程中并未展现的收获，其中涉及航天器和地面车辆，目前已在达拉斯/沃斯堡国际机场进行测试。

不管成本多大，测试仍需必不可少的认证，因为这是对产品本身，而不是抽象模型。然而，在某些情况下，测试成本可能会相当大。两个测试阶段（在形式化分析团队参与之前）都需要多方合作（机场官员、空中交通管理者、航空公司），以确保资源：两个机场跑道所预留的完整一天、飞行志愿者、测试评价工程师。此外，还需要一辆定制的厢式货车来模拟一架在地面上的飞机[53]。鉴于这些事实，可认为在更多产品中尽可能在较早的设计过程中包含形式化验证任务。

基于模型技术的优点是，除了成本很少以外，还更加全面。能够分析在模型中所有可能出现的情况，并发现以极小概率或在非常特殊条件下发生的潜在情况。这些几乎不可能在通常不超过 12 次测试飞行/天或模拟环节的测试程序中出现。与实际的状态空间尺寸相比（状态个数量级为 $10^{13} \sim 10^{42}$，这取决于参数的选择），这表明了详尽分析的必要性。形式化分析的另一结果是确定抽象模型中存在的问题，并提出消除这些问题的协议修改意见，从而显著提高设计正确性的可信度。

成功是由多种因素促成的：具有明确的规范以及设计工程师和形式化团队之间的良好沟通途径。另一方面，应用程序本身也有“正确”的大小（适用于整个状态空间结构）和“正确”混合连续状态分量和离散状态分量。

5.4.2 投入程度

RSM 的校验是一个持续 12 个月的中等规模项目。3 人小组共完成总计为 9 人—月工作量的不同阶段的分析工作。其中，大约 6 人—月的工作量主要用于决

策建模，在此期间，设计工程人员计划组织 3 次例会。最后的 3 人一月工作量用于验证本身，即制定查询、分析和综合答案。RSM 的理解在很大程度上得益于设计人员尽量保持规范和实现更加明确。总体而言，当人工努力主要体现在各个阶段的分析时，这主要是由于缺乏自动化建模过程以及建模人员对问题领域的陌生，而模型校验有效工具的影响对于本研究的完成至关重要。

5.4.3 故障容错

在上述 RSM 分析中没有涉及故障容错。尽管 RSM 操作正确性的验证都是假设在无故障条件下的，但数据链路故障的存在可能会显著影响算法的正确运行。这种类型的分析需要包括模型概率方面。对本研究的扩展自然而然地包括了良性性质（错过或延迟更新）或恶意/ Byzantine（参与方之间数据不一致）的故障行为。

5.4.4 面临挑战

除了模型校验实际应用中存在的习惯性挑战（减少巨大的状态空间，抽象合理性的争论），RSM 的经验还表现出一些新现象。首先，一个优秀的离散化方法和强大工具在处理这类复杂应用程序时是非常经典的。一些嵌入式应用程序可立刻推荐离散化规则。在这种情况下，算法每隔固定时间采集航天器位置，因此系统状态实际上已经离散化。消极的一面是实际的 C 代码仍无法验证。

在评估时态逻辑公式有效性方面，并不是所有类型的性能都能有效地验证。一般来说，对于安全性和可达性性能，存在较好的策略，而对于生存性和公平性，则需要较高的评估成本。

总的来说，可得出以下结论：在给定其能发现通常在仿真或测试阶段未显现出的重要错误情况下，模型校验是一种验证航空电子设备协议正确性的可行选择方案。

参 考 文 献

1. C. Adams, M. Consiglio, K. Jones, and D. Williams. SATS HVO Operational Concept: Nominal Operations, 2003.
2. R. Alur and D. L. Dill. Automata for modeling real-time systems. In *Automata, Languages and Programming, 17th International Colloquium, (ICALP)*, pp. 322–335, 1990.
3. T. Ball, R. Majumdar, T. Millstein, and S. K. Rajamani. Automatic predicate abstraction of C programs. *ACM SIGPLAN Notices*, 36(5):203–213, 2001.
4. S. O. Beskenis, D. F. Green, P. V. Hyer, and E. J. Johnson. Integrated Display System for Low Visibility Landing and Surface Operations. NASA Contractor Report 208446, NASA Langley, Hampton, VA, July 1998.

5. T. Bochot, P. Virelizier, H. Waeselynck, and V. Wiels. Model checking flight control systems: The airbus experience. In *ICSE Companion*, pp. 18–27, 2009.
6. R. E. Bryant. Graph-based algorithms for Boolean function manipulation. *IEEE Transactions on Computers*, 35(8):677–691, 1986.
7. P. Buchholz, G. Ciardo, S. Donatelli, and P. Kemper. Complexity of memory-efficient Kronecker operations with applications to the solution of Markov models. *INFORMS Journal on Computing*, 12(3):203–222, 2000.
8. T. Bultan, R. Gerber, and W. Pugh. Model-checking concurrent systems with unbounded integer variables: Symbolic representations, approximations, and experimental results. *ACM Transactions on Programming Languages and Systems*, 21(4):747–789, 1999.
9. J. R. Burch, E. M. Clarke, and D. E. Long. Symbolic model checking with partitioned transition relations. In A. Halaas and P.B. Denyer, eds., *International Conference on Very Large Scale Integration*, pp. 49–58, Edinburgh, Scotland, August 1991. IFIP Transactions, North-Holland.
10. V. Carreño and C. Muñoz. Formal analysis of parallel landing scenarios. In *Proceedings of the 19th Digital Avionics Systems Conference*, Philadelphia, 2000.
11. W. Chan, R. Anderson, P. Beame, S. Burns, F. Modugno, D. Notkin, and J. Reese. Model checking large software specifications. *IEEE Transactions on Software Engineering*, 24(7):498–520, 1998.
12. D. M. Chapiro. Globally-asynchronous locally-synchronous systems. PhD thesis, Stanford University, October 1984.
13. Y. Choi, S. Rayadurgam, and M. P. E. Heimdahl. Automatic abstraction for model checking software software systems with interrelated numeric constraints. In *Proceedings of the 9th ACM SIGSOFT Symposium on the Foundation of Software Engeneering (FSE)*, pp. 164–174, volume 26, 5, ACM Press, New York, September 10–14, 2001.
14. G. Ciardo. What a structural world. In R. German and B. Haverkort, eds., *Proceedings of the 9th International Workshop on Petri Nets and Performance Models (PNPM'01)*, pp. 3–16, Keynote paper, IEEE Computer Society Press, Aachen, Germany, September 2001.
15. G. Ciardo, et al. SMART: Stochastic Model Checking Analyzer for Reliability and Timing, User Manual. 2004. Available at: http://www.cs.ucr.edu/~ciardo/SMART/.
16. G. Ciardo, R. L. Jones III, A. S. Miner, and R. Siminiceanu. Logic and stochastic modeling with SMART. *Performance Evaluation*, 63(6):578–608, 2006.
17. G. Ciardo, G. Lüttgen, and R. Siminiceanu. Efficient symbolic state-space construction for asynchronous systems. In M. Nielsen and D. Simpson, eds., *Proceedings of the 21st International Conference on Applications and Theory of Petri Nets*, pp. 103–122, LNCS 1825, Springer-Verlag, Aarhus, Denmark, June 2000.
18. G. Ciardo, G. Lüttgen, and R. Siminiceanu. Saturation: An efficient iteration strategy for symbolic state space generation. In T. Margaria and W. Yi, eds., *Proceedings of the Tools and Algorithms for the Construction and Analysis of Systems (TACAS)*, pp. 328–342, LNCS 2031, Springer-Verlag, Genova, Italy, April 2001.
19. G. Ciardo, R. Marmorstein, and R. Siminiceanu. Saturation unbound. In H. Garavel and J. Hatcliff, eds., *Proceedings of the Tools and Algorithms for the Construction*

and Analysis of Systems (TACAS), pp. 379–393, LNCS 2619, Springer-Verlag, Warsaw, Poland, April 2003.

20. G. Ciardo, R. Marmorstein, and R. Siminiceanu. The saturation algorithm for symbolic state space exploration. *Software Tools for Technology Transfer*, 8(1):4–25, 2006.
21. G. Ciardo and A. S. Miner. A data structure for the efficient Kronecker solution of GSPNs. In P. Buchholz, ed., *Proceedings of the 8th International Workshop on Petri Nets and Performance Models (PNPM'99)*, pp. 22–31, IEEE Computer Society Press, Zaragoza, Spain, September 1999.
22. G. Ciardo and R. Siminiceanu. Using edge-valued decision diagrams for symbolic generation of shortest paths. In M. D. Aagaard and J. W. O'Leary, eds., *Proceedings of the Fourth International Conference on Formal Methods in Computer-Aided Design (FMCAD)*, pp. 256–273, LNCS 2517, Springer-Verlag, Portland, OR, November 2002.
23. G. Ciardo and R. Siminiceanu. Structural symbolic CTL model checking of asynchronous systems. In W. Hunt, Jr. and F. Somenzi, eds., *Computer Aided Verification (CAV'03)*, pp. 40–53, LNCS 2725, Springer-Verlag, Boulder, CO, July 2003.
24. E. Clarke, D. Grumberg, and D. Long. Model checking and abstraction. *ACM Transactions on Programming Languages and Systems*, 16(5):1512–1542, 1994.
25. E. M. Clarke and E. A. Emerson. Design and synthesis of synchronization skeletons using branching time temporal logic. In *Proceedings of the IBM Workshop on Logics of Programs*, pp. 52–71, LNCS 131, Springer-Verlag, London, UK, 1981.
26. M. Colón and T. E. Uribe. Generating finite-state abstractions of reactive systems using decision procedures. In *Proceedings of the Computer Aided Verification (CAV)*, 1998.
27. J. C. Corbett, M. B. Dwyer, J. Hatcliff, S. Laubach, C. S. Păsăreanu, Robby, and H. Zheng. Bandera: Extracting finite-state models from Java source code. In *International Conference on Software Engineering*, pp. 439–448, 2000.
28. P. Cousot. Proving the absence of run-time errors in safety-critical avionics code. In *EMSOFT*, pp. 7–9, 2007.
29. P. Cousot and R. Cousot. Abstract interpretation: A unified lattice model for static analysis of programs by construction or approximation of fixpoints. In *POPL*, pp. 238–252, 1977.
30. D. Dams, R. Gerth, G. Dohmen, R. Herrmann, P. Kelb, and H. Pargmann. Model checking using adaptive state and data abstraction. In *Proceedings of the 6th International Computer Aided Verification Conference*, pp. 455–467, 1994.
31. M. Davio. Kronecker products and shuffle algebra. *IEEE Transactions on Computers*, C-30:116–125, 1981.
32. G. Dowek, C. Muñoz, and V. Carreño. Abstract model of the sats concept of operations: Initial results and recommendations. NASA Contractor Report 213006, NASA Langley, Hampton, VA, March 2004.
33. B. Dutertre and M. Sorea. Modeling and verification of a fault-tolerant real-time startup protocol using calendar automata. In *Formal Techniques, Modelling and Analysis of Timed and Fault-Tolerant Systems, Joint International Conferences on Formal Modelling and Analysis of Timed Systems (FORMATS/FTRTFT)*, pp. 199–214, 2004.

34. S. Graf and H. Saïdi. Construction of abstract state graphs with PVS. In *Proceedings of the Computer Aided Verification (CAV)*, 1997.

35. D. F. Green, Jr. Runway Safety Monitor algorithm for runway incursion detection and alerting. NASA Contractor Report 211416, NASA Langley, Hampton, VA, January 2002.

36. J. Hatcliff, M. Dwyer, and S. Laubach. Staging static analyses using abstraction-based program specialization. *Lecture Notes in Computer Science*, 1490:134–151, 1998.

37. M. P. E. Heimdahl. Experiences and lessons from the analysis of TCAS II. In *Proceedings of the 1996 ACM SIGSOFT International Symposium on Software Testing and Analysis*, pp. 79–83. ACM Press, 1996.

38. C. Heitmeyer, J. Kirby, Jr., B. Labaw, M. Archer, and R. Bharadwaj. Using abstraction and model checking to detect safety violations in requirements specifications. *IEEE Transactions on Software Engineering*, 24(11):927–948, 1998.

39. D. Jackson, M. Thomas, and L. I. Millett. *Software for Dependable Systems: Sufficient Evidence?* National Academies Press, 2007.

40. D. Jensen. NASA's solution to runway incursion. *Avionics Magazine*, pp. 34–39, November 2003.

41. T. Kam, T. Villa, R. Brayton, and A. Sangiovanni-Vincentelli. Multi-valued decision diagrams: Theory and applications. *Multiple-Valued Logic*, 4(1–2):9–62, 1998.

42. S. Kimura and E. M. Clarke. A parallel algorithm for constructing binary decision diagrams. In *Proceedings of the International Conference on Computer Design (ICCD)*, pp. 220–223, IEEE Computer Society Press, Cambridge, MA, September 1990.

43. E. A. Lee. Computing needs time. *Communications of the ACM*, 52(5):70–79, 2009.

44. C. Livadas, J. Lygeros, and N. A. Lynch. High-level modeling and analysis of TCAS. In *Proceedings of the IEEE Real-Time Systems Symposium (RTSS)*, 1999.

45. Z. Manna, M. Colón, B. Finkbeiner, H. Sipma, and T. E. Uribe. Abstraction and modular verification of infinite-state reactive systems. In *Proceedings of the Requirements Targeting Software and Systems Engineering (RTSE)*, 1997.

46. K. L. McMillan. *Symbolic Model Checking*. Kluwer, 1993.

47. A. S. Miner. Efficient solution of GSPNs using canonical matrix diagrams. In R. German and B. Haverkort, eds., *Proceedings of the 9th International Workshop on Petri Nets and Performance Models (PNPM'01)*, pp. 101–110, IEEE Computer Society Press, Aachen, Germany, September 2001.

48. A. S. Miner and G. Ciardo. Efficient reachability set generation and storage using decision diagrams. In H. Kleijn and S. Donatelli, eds., *Proceedings of the 20th International Conference on Applications and Theory of Petri Nets*, pp. 6–25, LNCS 1639, Springer-Verlag, Williamsburg, VA, June 1999.

49. T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–579, 1989.

50. K. S. Namjoshi and R. P. Kurshan. Syntactic program transformations for automatic abstraction. In *12th Conference on Computer Aided Verification*, number 1855 in LNCS, 2000.

51. J. Souyris, V. Wiels, D. Delmas, and H. Delseny. Formal verification of avionics software products. In *FM*, pp. 532–546, 2009.

52. S. Tasiran, R. Alur, R. P. Kurshan, and R. K. Brayton. Verifying abstractions of timed systems. In *Proceedings of the International Conference on Concurrency Theory (CONCUR)*, 1996.

53. J. Timmerman. Runway Incursion Prevention System, ADS-B and DGPS data link analysis, Dallas—Ft. Worth International Airport. NASA Contractor Report 211242, NASA Langley, Hampton, VA, November 2001.

54. P. Wolper and B. Boigelot. Verifying systems with infinite but regular state spaces. In *Proceedings of the Computer Aided Verification (CAV)*, 1998.

55. Y. Zhao and G. Ciardo. Symbolic CTL model checking of asynchronous systems using constrained saturation. In Z. Liu and A. P. Ravn, eds., *Proceedings of the 7th International Symposium on Automated Technology for Verification and Analysis (ATVA)*, pp. 368–381, LNCS 5799, Springer-Verlag, Macao SAR, China, October 2009.

第 4 部分

通 信 系 统

第6章 形式化方法在有源网络通信服务中的应用

MarÍa Del Mar Gallardo、Jesús MartÍnez 和 Pedro Merino
编程语言与科学计算系，马拉加大学，马拉加，西班牙

6.1 简介

本章论述了形式化方法在通信服务开发以及有源网络方法的最新范例中的应用[7,21]。在该应用领域，已进行了协议的形式化和验证工作。但同时还发现具有更功能强大的方法可用于此目的。特别是，介绍采用一种更著名的工具（SPIN）并解释其具有的优势。本章的主要目的是提供一个在该应用领域中形式化方法的应用综述。另一个的目的是将本团队所提出的方法与之前的方法进行比较。

首先从描述解决有源网络和形式化方法的建议方法开始：Maude[12,28]、ActiveSPEC[23]、Unity[5]和 Verisim 框架[4]。然后，利用 SPIN 模型校验器[17,19]来描述最近的研究工作。

为公平起见，重点从3个主要方面来描述每种方法的优缺点：*有源网络的形式化建模*、*性能规范*及其*分析能力*。由于 SPIN 方法较为新颖，因此将用更多的篇幅来进行描述。

本章结构安排如下：6.2 节和 6.3 节将针对所讨论的问题领域进行简要概述；6.4 节给出一些相关文献中的经验；6.5 节侧重于 SPIN 显式模型校验器在示例建模和验证有源网络服务中的应用；最后，给出一些结论。

6.2 有源网络

目前通信网络的用户日益要求更新、更强的服务。如视频会议服务、实时处理、成组通信、移动或点对点服务等。在20世纪90年代开发有源网络的主要动机就是由于缺乏计算机网络与互联网相结合产生新协议和服务的灵活性。这种灵活性的缺乏至少有两个起因：一方面，标准化组织，如 ITU-T、IEEE 或 IETF 均需要一个漫长的过程来得到新协议的批准并在大多数网络节点和终端得到实现；另一方面，集成的新服务可能需要重新启动网络中的许多关键节点，并严重影响

数以百万计的用户。

有源网络的基本思想是开放某些节点的特定应用程序编程接口（API），以允许快速实现新的实验服务。自20世纪90年代初以来，一些研究团队就致力于定义节点架构、运行平台以及用于开发人员的描述语言。在这种背景下，值得关注的是IEEE所倡议的用于可编程网络接口的标准IEEE P1520，以及由国际信息处理联合会（IFIP）的第6技术委员会（TC6-通信系统）所组织的有源网络（IWAN）方面的一系列国际会议。

这种范例的引入在一些方面产生了可重用技术，如智能体、移动代码、中间件、操作系统、可重构硬件、路由器的数据包调度、路由协议、网络管理或安全代码注入与执行等。在实际应用方面，有源网络在诸如电子邮件[22]、视频分配[6,16]和覆盖网络与虚拟网络[25]等应用领域具有相当大的影响。

虽然实现一个开放互联网的最初兴趣现已逐渐减少，但所有这些技术都已在网络虚拟化，如Ad hoc网络、自主网络、传感器网络、内容感知网络等领域中得到新的发展机遇。

有关这些技术及其应用以及未来重用的完整描述请参见文献［20，21］。

6.3　CAPSULE法

一个有源网络允许引入新的服务，与通常遵循的较慢的标准化过程相比，该方法可更快地部署。引入新服务的一种方式是允许路由器和/或开关（节点）来执行可修改数据包内容、创建新数据包或甚至存储和读取节点信息的自定义计算。

在文献［2，27，30，31］中提出了主动节点的几种规划模型以及封装形式和处理的一些其他建议。所有现有平台的通信取决于两个标准。第一个是考虑不同执行环境（EE）的有源网络架构，以允许不同类型的节点[29]。在该方法中，一个有源网络节点的功能可分为执行环境和节点操作系统（NodeOS）两方面，分别负责提供网络抽象和管理网络资源的访问。如图6.1所示，该架构可允许在一个主动节点中存在多种执行环境，其中，由外界环境提供的表现力可能是一个完整的图灵机。NodeOS负责实现可访问节点资源的一组抽象行为。资源的访问应由安全执行引擎提供保护，即在执行关键任务之前请求代码认证。操作系统还可访问通信通道来发送和接收数据包，并受到数据存储功能的限制。

在这种新范例中，另一个重要标准是为非主动因特网上的主动包定义一个传输协议（ANEP）[1]。该协议用于支持全球范围的有源骨干网络。有源网络传输系统（ANTS）的程序设计模型构成一种最常用的有源架构，其中包括一个通过

移动代码技术在中间节点和终端系统中自动部署协议的 Java 工具包。可根据其特殊代码（有效负载）来处理称为封装体（capsules）的数据包。该代码可在节点采用称为 4 种基元的操作：*封装处理*（访问头和有效负载）、*控制操作*（允许封装体创建新的封装体，复制和丢弃自身）、*环境访问*（读取节点状态信息，如路由表）和*节点存储*（根据应用程序定义的目标来处理现有的存储）。每个封装体都是通过其类型和有效负载来识别（见图 6.2）。假设一种有源网络协议是由大量不同类型的封装体示例来实现的。

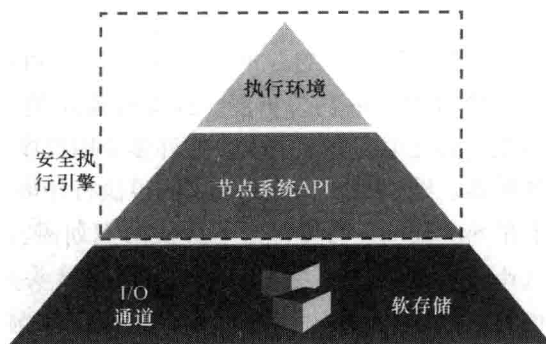


图 6.1　主动网络体系架构

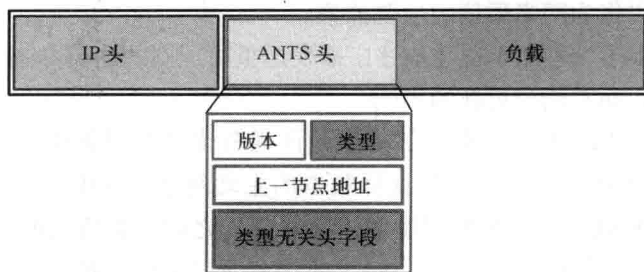


图 6.2　ANTS 以前使用的封装格式

在发送端设置处理封装体的代码，而不能在网络内部改变。封装体处理过程的能力有限，因为这是由不信任用户定义的。该封装体是由非主动节点通过路由信息转发的，但只在特定节点上执行代码。因此，在接收到封装体之后，主动节点才执行相关程序。封装在内部的代码用于决定是否继续转发给其目的节点，通常在处理代码的结尾处包含决策。这种代码传输的机制取决于具体实现。针对 ANTS，所提出的建议是根据要求来加载代码并通过缓存来提高性能。默认情况下，节点中的软存储只对具有相同协议的封装体共享。

6.4 有源网络的之前分析方法

本节重点综述了文献中有关采用形式化方法来描述和分析有源网络的现有方法。在此介绍的工具和方法有 Maude、ActiveSPEC、Unity 和 Verisim 框架。下节将着重介绍利用 SPIN 模型校验器来进行有源网络的模型校验。

6.4.1 Maude

Maude 是一种在大量应用程序中支持重写逻辑规范[24]和程序设计的反思型语言。支持逻辑反思可使得 Maude 显著扩展。这支持模块组成操作的可扩展代数运算，以允许采用先进的元编程和元语言。在许多应用程序中，Maude 用于创建不同逻辑、定理证明器、编程语言和计算模型的可执行环境。关于有源网络的规范，Maude 已用于在 Switchware 项目背景下开发的规划语言[28]操作语义下进行可靠性广播协议（rbp）[12]的建模。在 Maude 内，必须从头开始建立有源网络体系结构的规范。也就是说，必须使得节点、网络（节点实例及其关系）、执行环境和并发模型形式化，以及变量、表达式、语句、节点资源操作或路由策略的执行语言语义形式化。例如，在文献［12］中，必须构建一个消息传送的完整形式化基元，以作为网络通信的主要抽象。

图 6.3 给出了一种用于创建描述广播发送重写规则的建模方法。箭头左侧表示之前的情况，而右侧为更新的规则。

现在第 2 版本的 Maude 集成了一个允许规范化命题时态逻辑公式的模型校验器[13]。这用于在文献［28］所提低版本的主动网络配置中查找错误。遗憾的是，并没有详细阐述有关在采用模型校验器分析之前，如何对原始 Maude 模型的低版本进行研究的细节。文献［13］的深入研究表明该模块需要在更加实际的环境中进行进一步测试。

```
crl [Send] :
 < A : Node | nbs : OIDSET,
              states : states(state(A,passive),IDSTATUSPFUN),
              parents : IDIDPFUN,
              seqNos : seqNos(seqNo(A,N),IDINTPFUN),
              nbsStates : IDIDSTATUSPFUN >
=>
 < A : Node | states : assignIdStatusPFun(IDSTATUSPFUN, A, active),
              parents : assignIdIdPFun(IDIDPFUN, A, A),
              seqNos : assignIdIntPFun(IDINTPFUN, A, N + 1),
              nbsStates : assignAllIdIdStatusPFun(
                              IDIDSTATUSPFUN, A, OIDSET, active) >
 (multimsg N + 1 To OIDSET From A Src A)
if OIDSET =/= mtnbs .
```

图 6.3　Maude 中有关模型广播 ad hoc 的重写规则（源自文献［12］）

6.4.2 ActiveSPEC

ActiveSPEC[23]通过利用 PVS[9]定理证明器的先进功能来构成形式化规范主动服务和资源的框架。所提的该方法中包括定义数据包（见图 6.4）、节点、一组预定义服务、安全政策和资源。开发人员可利用定理证明器来验证在满足某些认证需要的执行环境中所处理的数据包。安全策略包括在执行数据包之前和之后所分别满足的前置条件和后置条件的定义。目前，这些规范是在按照框架建议的规范下手动输入的。图 6.4 给出了 ActiveSPEC 中表示数据包在使用前存在于队列中的前置条件下的封装体（数据包）定义。

```
Packets [
  Address : TYPE,      % Addresses
  PacketType : TYPE, % Packet type identifiers
  Payload : TYPE     % Payload data type
] : THEORY BEGIN

PacketID : TYPE+ Packet : TYPE = [#
  ptype : PacketType,
  src : Address,
  dest : Address,
  pid : PacketID,
  payload : Payload
#] packet_exists : AXIOM (EXISTS (pkt : Packet) : TRUE) IMPORTING
Queue[Packet]

END Packets
```

图 6.4　ActiveSPEC 中的封装体定义（源自文献［23］）

一个称为 ActiveNodeSPEC 的类似方法分析了在一个节点中资源的并发使用。所有这些框架都不涉及协议的安全性，这是因为均没有执行任何主动数据包内容的实现机制。由于侧重于不同方面，因此 ActiveSPEC 和 ActiveNodeSPEC 必须相互独立的使用。

该框架支持作为 PVS 定理的性能和安全策略的定义。因此，ActiveSPEC 利用通过需要交互验证的定理证明来作为其分析技术。

6.4.3 Unity

Unity[8]是一种允许进程组合的小型并行语言。语句的执行是以一种不确定形式根据所提供的语句来完成的，但不支持类型、控制流或隐式执行顺序。在文献［5］中该形式化方法用于主动网络协议建模。其中，设计了一个表示与主动平台和植入代码类型无关的主动节点输出的可编程能力的抽象接口。该接口建立了执行环境与执行代码之间的关系，并利用共享内存来建立通信模型（见图 6.5）。尽管与典型的基于迁移的语言有所不同，但认为 Unity 具有足够的灵活性以一种紧凑方式来表征程序，当植入代码满足某种限制时，可对主动节点的全局

状态测试提供一种基本支持。

```
Program {Node} Program at each active node v initially
  NO v.state, discCnt, errCnt = idle, 0, 0
assign
  N1 < < []x : v.inC[x] E v.inC :
   v.state, v.inC[x], v.Msg, v.LH := newPkt, tail(v.inC[x]), head(v.inC[x]), x >
   || < || i :: v.rt.i.usage := 0 )
   > if v.idle ^ (v.inC[x] != null)
...
```

图 6.5　Unity 中的节点定义部分（源自文献［5］）

关于性能，Unity 通过某些时态逻辑运算符具有定义进程和安全时空特性的功能。另外，还允许定义状态的前置条件和后置条件。在文献［5］中采用了手动方式证明 Unity 性能的方法，据目前所知，尚不具备对所提主动行为进行自动验证支持。

6.4.4　Verisim 法

Verisim 项目已开发出一套对网络仿真跟踪进行性能正确性分析的工具箱[4]。该方法采用多协议 ns2 模拟器的输出，这是一种通过 OTcl 脚本来配置节点、链接、协议堆栈并终止程序来执行网络仿真的知名理论工具。文献［26］中给出了作为一种扩展型 ns2 仿真器的主动服务配置的合适扩展。

Verisim 集成了元事件定义语言（Meta-Event Definition Language，MEDL），这是一种 ns2 执行跟踪定义模式的 LTI 扩展。通过定义跟踪上的事件和条件需求，MEDL 面向执行所谓的监测和校验（见图 6.6）。安全性从条件上必须如实描述且从不会产生报警。MEDL 利用辅助变量来记录执行历史。

```
alarm LoopInv[at][nxt][dst] = sendroute[at][dst] when
    ((at!=nxt) && (at!=dst) && (nxt!=dst) &&
     (obs_nexthop[at][dst] == nxt) &&
     ((obs_seqno[at][dst] > obs_seqno[nxt][dst]) ||
      (obs_seqno[at][dst] == obs_seqno[nxt][dst]) &&
      (obs_seqno[at][dst] <= obs_seqno[at][dst])))
```

图 6.6　MEDL 中描述性能的示例（参见文献［4］）

Verisim 技术可认为是一种自动的运行时分析技术。因此，不必在模型中进行详细分析。

6.5　SPIN 有源网络模型校验

在过去的几年中，SPIN 已成为一个在学术和工业领域中广泛采用的模型校验器[19]。SPIN 支持验证以建模语言 PROMELA 编写的系统中的常见安全性能（如无死锁）以及以线性时态逻辑表示的复杂性需求分析。另外，也用作解决状

态爆炸问题的功能强大的新算法的测试平台。

PROMELA 是一种用于描述由并发异步通信进程组成的系统的语言。一个 PROMELA 模型包含一组进程 $M = \mathrm{Proc}_1 \parallel \cdots \parallel \mathrm{Proc}_n$、全局通道和局部通道、全局变量和局部变量的有限集合。通过通信通道，进程之间利用消息传递进行通信。通信可以是通过将通道作为有界缓冲区进行异步通信，也可以是通过零尺寸大小的通道进行同步通信。全局通道和全局变量决定了进程运行的环境，而局部通道和局部变量则建立了各进程中的内部局部状态。PROMELA 处理可定义为一系列在声明部分之前的可能标记的语句。同时，还提供了用于表示不确定性执行的机制。

SPIN 模型校验器可验证 PROMELA 模型的线性时态逻辑（LTL）公式。符合语法规则的公式是根据一组原子命题（在 PROMELA 中，命题是数据、通道或标签上的测试）、标准的布尔运算符以及时态算子（*总是*“□”、*最终*“◇”和*直到*“U”）归纳并构建的。可通过模型状态序列（跟踪轨迹）$t_i = s_i \rightarrow s_{i+1} \rightarrow \cdots$对公式进行解释。每个序列表示一个从状态 s_i 开始的可能的模型执行。采用时态算子可允许构建取决于配置跟踪轨迹中当前状态和未来状态的公式。

默认情况下，给定一个 LTL 公式，SPIN 可将其转换成一个表示不可取行为（认为不可能）的 Büchi 自动机。然后，对全面搜索满足自动机执行的状况空间进行验证。如果存在某种执行，那么该工具就将其记录为性能反例。如果搜索整个模型而没有发现一个反例，则认为该模型满足 LTL 性能的*通用属性*。同样的验证方案还可以用来检查是否一个公式不能满足所有路径（存在性反驳）。在图形用户接口中显示的两种 LTL 使用方式称为 XSPIN。

尽管已有许多可以进行详尽标准验证的示例，SPIN 还实现了处理复杂系统的优化技术。*部分降阶*利用只表示整个集合的事件序列来代替几个交叉序列的事件（语句）。*状态压缩*可通过不丢失信息的压缩状态表示来减少内存占用。*位状态散列*以哈希表形式来表示状态，因此在许多情况下，都只是部分分析。

6.5.1　PROMELA 中的有源网络模型

在此，考虑一个如何在 PROMELA 中构建类似 ANTS 的主动协议模型的示例，需要注意的是，验证模型应尽可能小以确保其能表示性能分析所需的具体细节。在本例中抽象的基本元素是①网络链接和拓扑；②主动 NodeOS 和③在主动节点内执行的主动应用程序，以及端到端应用程序。这些元素如图 6.7 所示。因此，生成模型将会有一部分表示主动节点（操作系统工具 + 执行环境），另一部分表示新的主动应用程序和拓扑结构。对于每个主动配置，前者非常普遍，而后者将会根据所要求的主动服务而改变。

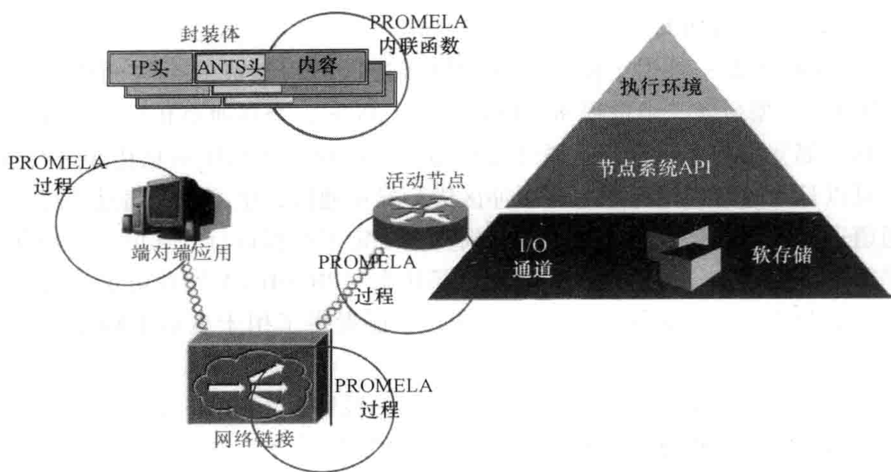


图 6.7　一个主动网络的 PROMELA 模型概述

6.5.1.1　封装体建模

在主动式 ANTS 的专业术语中，一个封装体是一个在 PROMELA 中建模为新数据类型的主动数据包，如图 6.8 所示。关于图 6.2 中所示的数据包字段，PROMELA 中只包含 4 个强制字段：源地址和目的地址，执行封装体的上一节点，当到达一个新节点时将要执行的相关参考代码。如果在开发新的主动协议时需要，则基本的封装体类型可扩展到特定字段。

```
typedef Capsule{
    address src;  /*source address of the IP datagram*/
    address dst;  /*destination address of the IP datagram*/
    address prev; /*previous active node. Reserved to Active Nodes*/
    codeRef ref;  /*type of capsule (it makes reference to an active application code)*/
    /*Specific data fields must be inserted from here*/
};
```

图 6.8　一个类似 ANTS 封装体的 PROMELA 定义

6.5.1.2　共同部分：节点、通信和环境

在 PROMELA 中，通信可利用异步通信通道并根据系统初始化时设计人员提供的拓扑信息通过在网络中执行数据包交换来建模。通过该过程，主动路由器并不关心具体的路由处理工作，而会减少验证阶段存储状态的交叉和规模。图 6.9 中所示的 *AbstractNetworkSerive* 是根据 ANTS 工具包的无连通特性对于不可靠通信的一个基本传输工具[29]。当接收到一个新的封装体时，检查其目的字段。如果目标网络与发送属于主机/节点的封装体为同一网络，则网络服务可直接传输封装体。但是，如果目标网络为模型中的另一个网络，则网络服务必须检测是从路由器还是从主机接收到封装体。如果封装体是来自路由器，则必须查询其路由信

息以选中执行下一个 hop 的路由器。如果封装体来自主机，则网络服务将其转发给本地网络路由器。

```
active proctype AbstractNetworkService(){

    Capsule capsule;
    address from;
endService:
    do
    :: atomic{ to_network ? from,capsule->
        if
        /*packet destined to the same network*/
        :: (getNetworkFromAddress(from))==(getNetworkFromAddress(capsule.dst))->
            if
            ::(getHostFromAddress(capsule.dst)==0)-> /*router*/
                    from_network[getNetworkFromAddress(capsule.dst)]!capsule;
            ::else-> /*host*/
                    from_network[getHostFromAddress(capsule.dst)]!capsule;
            fi
        /*packet destined to another network from a Host*/
        ::((getNetworkFromAddress(from))!=(getNetworkFromAddress(capsule.dst)))
          && (getHostFromAddress(from)!=0) ->
                    from_network[getNetworkFromAddress(from)]!capsule;

        /*packet destined to another network from a Router*/
        ::((getNetworkFromAddress(from))!=(getNetworkFromAddress(capsule.dst)))
          && (getHostFromAddress(from)==0)->
                    from_network[NODE[getNetworkFromAddress(from)].route]!capsule;
#if WITH_LOSSES
        ::skip
#endif
        fi;
      }
    od
}
```

图 6.9　PROMELA 中的不可靠网络通信服务

主动路由器可作为进程进行建模，以允许执行主动应用程序。对这些 PROMELA 中的实体进行抽象构成了一个包含 NodeOS 和执行环境的混合体。一个重要的设计要求是假定在系统启动时已在主动节点中加载了主动应用程序，因为代码到达节点的方式是独立于所要分析的主动服务的。例如，图 6.10 所示的执行环境中包括从主动网络接收封装体数据包并执行与参考代码相关的对应主动应用程序所需的代码。由图中可知，*from_network_to_i* 的接收通道已定义为只读，在 SPIN 中的一种优化方法是使得验证中的状态爆炸影响最小。当接收到一个封装体时，检查其引用类型字段并且在执行环境中选择处理封装体的适当程序。如图 6.10 所示的示例中，存在 3 种类型的封装体。

为处理封装体，现有几种函数来获得其基本头字段（getSrc，getDst，gelPrev）。setDst 和 newCapsule 函数分别改变封装体的目的地址和创建一个新的转发地址。其他函数包括显式转发和路由以及负责处理节点中软存储数据缓存的操作基元（cacheGet，cachePut，cacheRemove），正如在 ANTS 文献中所采用的[30]。

```
proctype EE(address _this_address){

    chan from_network_to_i = from_network[getNetworkFromAddress(_this_address)];
    xr from_network_to_i;
    Capsule capsule;
endEE:
    do
    :: atomic{ from_network_to_i?capsule ->
                if /*time to process capsules*/
                /*Code for processing active applications will be inserted here*/
                ::(capsule.ref==DATA)-> capsuleDATA();
                ::(capsule.ref==NACK)-> capsuleNACK();
                ::(capsule.ref==ACK)-> capsuleACK();
                ::else-> /*references that were not previously registered*/
                     if
                     ::(capsule.dst == _this_address)-> /*silently discarded*/
                     ::else-> to_network!_this_address,capsule;
                     fi
                fi;
       }
    od
}
```

图 6.10　一个执行环境的示例

6.5.1.3　创建网络拓扑结构

当部署一个新的网络服务时，主动网络设计人员应处理网络拓扑结构。为进行验证而抽象一个真正的拓扑结构，PROMELA 拓扑结构由各个网络组成，其中每个网络都由自然数标记。基本的连接机制是网络之间的点对点连接，每个连接中都包含一个主动节点。因此，这些主动节点的标识符（全局地址）与那些分配到相应网络中的一样，从而避免了需要一些其他的复杂路由机制。主动主机执行隶属于网络的端到端应用程序。其全局地址由两个不同部分组成：所属网络和一个有效的主机标识符。主机标识符是从在定义网络时之前未用的下一个有效值开始的自然数集中选择的。*hostAddress*（*idNet*，*idHost*）函数可从有效网络和主机标识符中自动获取所得的全局地址。图 6.11 给出了一个由两个网络组成且每个网络都具有相应的主动节点（标记为 0 和 1）的主动网络拓扑结构。

```
/* Network topology for verification:                    /--receiver(3)
                                  sender(2)---n(0)---n(1)-
                                                         \--receiver(4)
*/
init{
 atomic{
    NODE[0].route = 1;
    NODE[1].route = 0;

    address addrRcv = hostAddress(1,3);
    ...
    start_router(0); /*n0*/
    start_router(1); /*n1*/
 }
}
```

图 6.11　PROMELA 中创建的网络拓扑结构

当启动系统时，设计人员必须提供这种拓扑信息。也就是说，在模型中必须明确声明每个主动节点的路由。最后，利用 *start_router*（*idNet*）函数来启动主动节点的发送器和接收器。

6.5.2　实例：验证主动协议

图6.8～图6.11对应于验证主动组播协议的示例部分[14]。称为RMANP的所选协议[6]构成主动网络中组播应用程序的一个复杂协议。值得注意的是，这是文献［3］中所示的可靠组播协议（RMNP）的面向ANTS版本。RMANP充分利用了使得主动节点参与处理确认（ACK）和重传请求（NACK）消息的主动网络。将多个ACK聚合为一个单一封装体（MACK）的来源可防止内爆的确认。当请求重新传送特定数据到目标应用程序时，主动节点中存储的不确定数据可保证传输通畅。节点还可过滤信息源的NACK消息，只允许那些请求的新数据通过。对于服务质量，RMANP考虑3种通信量：可靠性、不可靠性和具有时态约束的可靠性。文献［14］中提出的PROMELA模型只包括建模最复杂的可靠型服务的合适封装体（DATA、ACK和NACK）对。因此，在这些封装体中的主动代码只包括用于验证协议中性能的行为。利用图6.1所示的用于模型校验的拓扑结构，第一个网络中包括数据源：利用滑动窗口协议向主动节点提供数据的发送方。第二个网络由一组建模组播接收的进程构成。

作为一个封装体的代码示例，图6.12给出了该协议下的一个NACK封装体示例。主动节点拦截一个分析NACK序列字段的重传请求，只要在其高速缓存内能够提供可用的数据请求。将数据输入到缓存中的操作对应于DATA封装体的执行（更多细节参见文献［14］）。

```
typedef Capsule{
    address src;
    address dst;
    address prev;
    codeRef ref;
    byte seq;    /*RMANP specific field*/
    mtype data; /*payload*/
};

inline capsuleNACK(){
 short tmp;
 tmp = cacheGet(last_unack);
 if
 ::(tmp==capsule.seq)-> //send data from cache
   capsule.data = cacheGet(storing_key(tmp));
   sendCapsuleTo(getSrc(),DATA)
 ::else -> sendTo(getDst()) //forward to source
 fi
}
```

图6.12　RMANP的一个NACK封装体

对于RMANP，PROMELA中主动网络架构的具体部分和固定部分是无死锁的。此外，封装体的代码是用于利用SPIN来验证某些LTL公式的。例如，图6.13中的LTL性能表明一个必须满足所提RMANP服务中任一可能情况的通用属性。公式中的每个命题（以#define标记）都对应于协议封装体中主动代码的标记迁移。

```
LTL property (RestrictedScope):  [](retransmission_requested -> <> data_from_EEs)

  #define retransmission_requested   receiver[1]@retransm_req
  #define data_form_EEs    (data_from_EE0 || data_from_EE1)
  #define data_from_EE0    (EE[1]@retransm_EE_from_nack || EE[1]@retransm_EE_from_ack)
  #define data_from_EE1    (EE[2]@retransm_EE_from_nack || EE[2]@retransm_EE_from_ack)
```

图 6.13　通过 RMANP 的 SPIN 版本进行通用属性验证

6.5.3　在 SPIN 中更实际的代码建模

采用任何形式化描述（包括 PROMELA）的一个主要问题是需要手动抽象将要验证的应用程序的行为和结构。实际上，这种方法非常容易出错。因此，通常需要一个指导从源代码中的模型自动抽象（验证）方法[10、11、15]。在主动网络领域，这不仅涉及封装体的 Java 代码解析和转换，还提供主动节点 API（接口和行为）和 IP 路由算法的抽象表示。

为便于执行这些任务，最近的 SPIN 中集成一些扩展功能以允许在 PROMELA 内执行嵌入式 C 代码。根据该模型校验的新功能，现在就可以通过更实际的行为来丰富验证模型，从而提高所得结果的信任度。

图 6.14 给出了之前 PROMELA 网络模型（见图 6.9）的一种可能修改。在新的网络模型中，利用新的 c_ decl 基元来嵌入模型中新的 C 数据类型。在此，描述了对应于 PROMELA 中之前简化版本（现在正确重命名为 pml_ NodeResources）的 c_NodeResources 类型。因此，可处理更复杂的抽象数据类型，如路由表（具有目的网络和输出接口两列的二维阵列）或*最近最少使用缓存*（图中并未显示，包含在一个外部 . h 文件中）。在本例中，现在可允许路由器具有两个以上的接口，从而允许有比图 6.11 所描述的更复杂的拓扑结构。

C 变量可在模型中利用显式附属于状态向量或隐式从状态向量中提取的 c_ state 基元来声明。在图 6.14 所示的示例中，声明一个名为 nodeRes [ ] 的全局数组。同时，还可以声明一个在 PROMELA 原型本地范围内的 C 变量。

以不可分方式执行纯 C 代码可在 c_ expr 和 C 代码基元中实现。前者需要一个对于 C 代码段的无不良影响的布尔表达式，而后者是一个允许对 PROMELA 或 C 变量操作的更一般基元。图 6.14 表明了如何在网络服务原型中使用 c_ code，以在路由转发算法中包含更实际的行为。值得注意的是，利用前缀 *now* 来访问 C 代码段中的 PROMELA 变量。对于全局变量和原型（名为 proctype_ name）范围内变量的 Pproctype_ name- >。在该图的示例中，利用一个外部的 C 函数（getROUTE（））来计算转发封装体的合适输出通道。

总之，这些 SPIN 的扩展功能有助于用户创建 PROMELA 模型，使之更接近于试图验证的最终代码。采用模型中的 C 函数和算法具有重大意义，这是因为这些都是自动执行的。然而，通常还需要具有管理验证中的更多数据，这可能会

导致产生状态爆炸问题。

```
/*Promela version of a the NodeResources type*/
typedef pml_NodeResources{
address route; /* only an output interface is allowed*/
short cache[STORAGE_RESOURCES]=FREE;
/*Beware thinking in Cache as an LRU one. We have modeled it as a Hash Table instead,
  so be careful with the STORAGE_RESOURCES limit and your special needs*/
};

/*C version of a the NodeResources type*/
c_decl{
#include "nodetypes.h"
  typedef struct{
    short myAddress; /*node address of this node*/
    short routeTable[MAX_ROUTES][2];
    /*1st column: destination address, 2nd: output interface (channel id)*/
    struct LRU cache;
    /*now this cache is an array with extra information to get the Least Recently Used index*/
  }c_NodeResources;
}

c_state "c_NodeResources nodeRes[TOTAL_NODES]" "Global" /* goes inside the state vector */

/*C version of NetworkService (not as abstract as before) */
active proctype NetworkService(){

    Capsule capsule;
    address from;
    short output_channel_id ;
endService:
    do
    :: atomic{ to_network ? from,capsule->
        c_code{
            PNetworkService->output_channel_id = getRoute(PNetworkService->from,PNetworkService->capsule.dst);
            printf("Capsule will be forwarded to: %d\n", now.nodeRes[PNetworkService->output_channel_id].myAddress);
        }
        if
        ::(output_channel_id != -1) ->
            if
            ::from_network[output_channel_id]!capsule;
            #if WITH_LOSSES
            ::skip
            #endif
            fi
        ::else /* forwarding error ...*/
        fi;
      }
    od
}
```

图 6.14　在 SPIN 中嵌入 C 代码

6.6　小结

采用 PROMELA 作为主动协议和线性时态逻辑的形式化建模方式来描述其性能具有比之前方法更大的优势。关于*形式化规范*，在 6.4 节中介绍的语言都不能用于描述通信协议。因此，这些语言都不足以满足协议工程人员所采用的语言，其中，通常需要考虑基于迁移的语言。PROMELA 集成了协议开发所需的功能：消息、通信通道、损耗以及直接为特定主动网络架构提供支持的重新排序或并发模型。此外，PROMELA 通常与执行最终代码（Java 或 C）时的命令式语言具有许多相似性。与某些其他形式化方法（如 Maude 或 Unity）相比，这使得更加易于编写 PROMELA 代码，从而能够实现从最终源代码中抽象有效（自动）模型。

在性能方面，LTL 可支持除 ActiveSPEC 之外所介绍的几乎所有方法。值得

注意的是，采用未扩展的时态逻辑（如在 MEDL 中使用辅助变量）具有更高的效率。SPIN 还可采用线性时态逻辑运算符的整个范围，而不是 Unity 中缩减了的那部分。此外，对于不熟悉逻辑运算的用户，SPIN 利用图形化的线性时间编辑器工具[18]或嵌入式观测器（同步自动机）提供了一些表示性能的辅助逻辑符号。

最后，最实际的分析工具是可以执行自动验证的，即对所有可能的执行方案进行完备搜索。

参考文献

1. D. S. Alexander, et al. Active Network Encapsulation Protocol (ANEP). 1997. Available at: http://www.cis.upenn.edu/~switchware/ANEP/.
2. D. S. Alexander, et al. The Switchware active network architecture. *IEEE Communications Magazine*, 1998.
3. A. Azcorra, M. Calderon, and M. Sedano. A strategy for comparing reliable multicast protocols applied to RMNP and CTES. In *IEEE Conference on Protocols for Multimedia Systems—Multimedia Networking (MmNet'97)*, 1997.
4. K. Bhargavan, C. A. Gunter, M. Kim, I. Lee, D. Obradovic, O. Sokolsky, and M. Viswanathan. Verisim: Formal analysis of network simulations. *IEEE Transactions on Software Engineering*, 28(2):129–145, 2002.
5. S. Bhattacharjee, K. Calvert, and E. Zegura. Reasoning about active network protocols. In *IEEE ICNP'98*, 1998.
6. M. Calderon, M. Sedano, A. Azcorra, and C. Alonso. Active network support for multicast applications. *IEEE Network*, 12(3):46–52, 1998.
7. K. L. Calvert, S. Bhattacharjee, E. Zegura, and J. Sterbenz. Directions in active network research. *IEEE Communications Magazine*, 36(10):72–78, 1998.
8. K. Chandy and J. Misra. *Parallel Program Design*. Addison-Wesley, 1998.
9. J. Crow, et al. A tutorial introduction to PVS. Presented at *WIFT'95*, 1995.
10. P. de la Camara, M. M. Gallardo, P. Merino, and D. Sanan. Checking the reliability of socket based communication software. *STTT*, 11(5):359–374, 2009.
11. P. de la Camara, R. J. Castro, M. M. Gallardo, and P. Merino. Verification support for ARINC-653 avionics software, software testing, verification and reliability. *Software Testing, Verification and Reliability*, 21(4):267–298, 2011.
12. G. Denker, et al. Specifying a Reliable Broadcasting Protocol in Maude. Internal Report, Computer Science Laboratory, SRI International, Menlo Park, CA, 1999.
13. S. Eker, J. Meseguer, and A. Sridharanarayanan. The Maude LTL model checker. *Electronic Notes in Theoretical Computer Science*, 71, 2002.
14. M. M. Gallardo, J. Martinez, and P. Merino. Model checking active networks with SPIN. *Computer Communications*, 28:609–622, 2005.
15. K. Havelund and T. Pressburger. Model checking Java programs using Java pathfinder. *STTT*, 2(4):366–381, 2000.
16. D. He, G. Muller, and J. L. Lawall. Distributing MPEG movies over the internet using programmable networks. In *Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS'02)*, 2002.

17. G. J. Holzmann. The model checker SPIN. *IEEE Transactions on Software Engineering*, 23(5):279–295, 1997.
18. G. J. Holzmann. *The SPIN MODEL CHECKER. Primer and Reference Manual*. Addison-Wesley, 2003.
19. G. J. Holzmann. On-the-fly LTL Model Checking with SPIN. Available at: http://spinroot.com/spin/whatispin.html
20. S. A. Hussain. *Active and Programmable Networks for Adaptive Architectures and Services*. Auerbach Publications, 2006.
21. D. Hutchison, S. Denazis, L. Lefevre, G. Minden, and J. Gary. Active and programmable networks. In *IFIP TC6 7th International Working Conference*, IWAN 2005, Sophia Antipolis, France, Springer-Verlag, November 21–23, 2005. Revised Papers, 2009.
22. S. Kamouskos and A. Vasilakos. Active electronic mail, in SAC. 2002.
23. C. Kong, D. Dieckman, and P. Alexander. Formal Modeling of Active Network Nodes Using PVS. *Workshop on Formal Methods in Software Practice (FMSP-00)*, 2000.
24. J. Meseguer. Conditional rewriting logic as a unified model of concurrency. *Theoretical Computer Science*, 96:73–155, 2002.
25. N. M. Mosharaf Kabir Chowdhury and R. Boutaba. A Survey of Network Virtualization, University of Waterloo, Ontario, Canada. Technical Report CS-2008-25, 2008.
26. G. Rodriguez, P. Merino, and M. M. Gallardo. An extension of the ns simulator for active network research. *Computer Communications*, 25:189–197, 2002.
27. B. Schwartz, et al. Smart packets for active networks. *ACM Transactions on Computer Systems*, 18(1):67–88, 2000.
28. M. Stehr and C. Talcott. PLAN in Maude. Specifying an active network programming language. *Electronic Notes in Theoretical Computer Science*, 71, Elsevier, 2002.
29. D. Tennenhouse and D. Wetherall. Towards an active network architecture. *Computer Communication Review*, 26:2, 1996.
30. D. J. Wetherall, J. V. Guttag, and D. L. Tennenhouse. ANTS: Network services without the red tape. *IEEE Computer*, 1999.
31. Y. Yemini and S. da Silva. Towards programmable networks. In *IFIP/IEEE International Workshop on Distributed Systems: Operations and Management*. L'Aquila, Italy, October, 1996.

第 7 章　通信协议概率模型校验的实际应用

Marie Duflot
洛里亚，团队模型/VERIDIS，南锡，法国
Marta Kwiatkowska
计算机科学系，牛津大学，牛津，英国
Gethin Norman
计算机科学系，格拉斯哥大学，格拉斯哥，英国
David Parker
计算机科学学院，伯明翰大学，伯明翰，英国
Syl Vain Peyronnet
LRI，INRIA 巴黎第十一大学，奥赛，法国
Claudine Picaronny
LSV，法国科学研究中心，卡尚，法国
Jeremy Sproston
计算机系，意大利都灵大学，都灵，意大利

7.1　简介

如今，计算机控制设备普遍存在于许多行业领域，包括商业和安全关键领域。为此，采用设计于管理设备之间相互作用的通信协议是非常普遍的。近年来，在形式化验证方法的开发和配置方面具有显著发展，这主要用于建立大量实际系统中故障辨识的正确性，事实证明了涵盖本书所讨论主题的广度。

本章介绍了将形式化验证技术应用到通信协议的研究工作。这种系统代表了在形式化验证方面特别具有挑战性的情况，为成功建模和分析，必须考虑以下几个关键方面：多组元之间的*并发性*，在不同位置和未知速度下的可能操作；在协议以及所设计的操作媒介施加的准确*实时*约束；通常用于破坏相同协议下设备通信对称性的*随机性*。

*概率模型校验*是一种分析系统中随机行为的形式化验证技术。该技术首先构建某个实际系统的概率模型，并根据该模型的数学分析来确定原始系统的有用特性。与传统的验证技术不同，概率模型校验不仅可以用来确定正确性，也可用于确定性能和可靠性等定量分析措施。这些技术可应用于一系列的概率模型，通常

是各种形式的马尔可夫链，但在此特别重要的是在概率时间自动机（PTA）方面的应用，这是一种可巧妙地表征不确定性、实时性和概率行为的模型。本章对 PTA、概率模型校验和相应的实现技术与工具进行概述。此外，还通过一个案例分析（用于以太网网络中实例的 IEEE 802.3（载波监听多路访问/冲突检测［CSMA /CD］）协议）来阐述其在通信协议领域的作用。

7.2 PTA

PTA 是一种对具有不确定性、实时性和概率特征行为的系统进行形式化建模的方法[25]。PTA 是一种扩展的时间自动机[1]，这是一种对实时系统形式化验证的最具发展前景的形式化方法。

本节通过图 7.1 所示的示例来阐述（概率）时间自动机的一些基本概念。图中给出了一个站点试图在总线上传输的简单通信协议的自动建模。如果某个站点的传输被另一个站点的传输中断（冲突），那么该站点就停止活动，并在再次开始发送信息之前等待一个随机产生的时间。模型的*控制状态*（传输、等待和完成）可通过图的节点以及图边缘表示的控制状态之间的可能转移来表征。在初始状态，通过双向边界表示的传输是指系统进行的传输。从开始传输的 5 个单位时间后，消息成功发送，且模型移动到完成状态。时间依赖性可通过边的*时钟*、*值守*和*重置*以及状态的*不变量*来表征。时钟是一个实数变量且系统开始执行时为 0，当模型位于同一控制状态时以与实际时间相同的变化速度增加。该示例模型中具有单个时钟 x。从 TRANSMIT 到 DONE 的边的值守为 $x=5$，且当时钟 x 的值为 5 时表明可遍历边。TRANSMIT 状态的不变量是 $x\leqslant 5$，且表明在时钟值 x 大于 5 之前，控制必须通过 TRANSMIT。

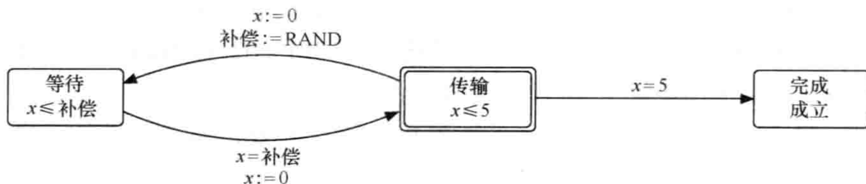


图 7.1　一个概率时间自动机示例

从 TRANSMIT 到 WAIT 的边并未由值守标记，这可解释为从 TRANSMIT 总是可遍历边，而无论时钟 x 为何值。不管是否采用该边，在自动机位于 TRANSMIT 状态内的任何时间点都是一种*不确定性*选择。该边的遍历对应于网络中其他站点同时在总线上传输所引起的中断。边具有由 $x:=0$ 表示的时钟复位功能，这表明在进入 WAIT 状态时，时钟 x 重置为 0。除了时钟，还允许称为重置、值

守和不变量的有限域变量，此外还允许这些变量的概率重置，相对于时钟重置时确定赋值为0。在遍历从TRANSMIT到WAIT的边时，根据赋值补偿：=RAND（其中，RAND是一个可能补偿值的概率分布，如自然数3~10的均匀分布），有限域变量补偿值可设置为一个随机值。另外，随后将在状态WAIT的不变量以及从WAIT到TRANSMIT的边的值守中使用该*补偿*值，以确保当时钟x的值达到补偿值时，可控制返回到状态TRANSMIT。

图7.1给出了一个PTA的示例。值得注意的是，通过标准时间自动机上的非概率赋值结果代替了概率赋值补偿：=RAND。虽然并未在该示例中阐述，PTA还可由一组标记边的*事件*进行扩展。然后，这些事件可用来定义一些PTA的*并行复合*，其中，每个自动机都看作整个系统的一个子组件，同时，自动机在共享事件上同步。这种事件将在7.4节中根据CSMA/CD协议来定义总线的PTA并行复合和两个站点的操作。在文献［27］中基于时间自动机的案例[1]和概率转移系统[33]引入并行复合算子。

PTA的语义可正式定义为一个无限状态马尔可夫决策过程（MDP），这是一个支持不确定性选择的概率模型，且以离散节拍进行转移。由于PTA的时钟为实数变量，因此MDP通常包含无限状态。同理，随时间变化的状态转移在数量上也是无穷的，这是因为这对应于实数时间间隔。通常由时态逻辑中规范的一个或多个性能的形式化验证的PTA分析是非常重要的，因为模型校验算法通常用于有限状态模型。然而，根据（非概率）时间自动机模型校验的类似研究，可获得PTA的准确可靠的有限状态表示，这可用于分析大范围性能正确性或性能指标。这种表示的一个示例是基于*数字时钟*，其中PTA的时钟取为整数值，而不是实数值[23]。这种方法要求值守时钟和不变量时钟的对比不是很严格，然而这是实际系统的PTA模型中的常见性能，如本章考虑的CSMA/CD协议。到目前为止，数字时钟的方法已成功用于所提出的PTA有限状态表示实践中：即已用于验证CSMA/CD协议[10,28]、部分无线局域网的IEEE 802.11标准[26,31]、相线根（FireWire root）争用协议[27]和IPv4的零配置网络协议[23]。

另一种PTA的分析方法已基于*区域*操作进行开发，即凸多面体。在文献［25］中介绍了基于状态空间前向遍历的该方法。随后在文献［22］中的工作中通过采用抽象细化和随机游戏扩展了其适用范围。另一方面，文献［28］中开发的技术是基于状态空间的后向搜索，现已用于成本和回报性能的分析[5,12]。

7.3 概率模型校验

概率模型校验是一种用于具有随机行为的系统建模和分析的形式化验证技术。该方法可应用于几种不同类型的概率模型中。3个最常见的应用是：时间建

模为离散节拍且离散概率选择随机性的离散时间马尔可夫链（DTMC）；将 DT-MC 扩展具有不确定性行为表征能力的 MDP；不允许非确定性但允许根据指数分布规范实际（连续）时间特性的连续时间马尔可夫链（CTMC）。本章主要关注 MDP，这是因为上节所述的许多情况下 PTA 的分析都可归类为 MDP 的分析。另外，也讨论了 DTMC，这可看作一种不存在不确定性的特殊 MDP。

DTMC 和 MDP 的性能通常都以时态逻辑表示，如概率计算树逻辑（PCTL）[6,15]和线性时态逻辑（LTL）[9]。根据这些逻辑，可推导以下定量：

- “协议最终终止的概率”；
- “每个请求后确认或超时信号的概率”；
- “在 35μs 内成功接收消息的概率”。

在 MDP 情况下，实际上不可能计算得到精确的概率值，这正是由于在模型中存在不确定性。在这种情况下，所计算的值为所有可能不确定性结果中的*最小*概率或*最大*概率。这对应于*最好情况*或*最坏情况*的分析（根据概率含义）。此外，这也常常适用于 DTMC 和 MDP，来计算模型整个可能配置范围或参数上的最小概率或最大概率，以产生一种不同的最好/最坏分析。最后，值得注意的是，还可以通过实数值的*成本*或*回报*来增强 DTMC 和 MDP，这可由多种度量方式来表示，例如，“系统电能损耗”、“消息丢失数量”或“消息队列大小”。然后就有可能推导出（可能是最低或最高）这些措施的期望值，例如：

- “系统操作前 20s 的预期电能损耗”；
- “协议终止前的预期消息丢失数量”；
- “500μs 后预期发送消息队列数量”。

7.3.1 概率模型校验技术

7.3.1.1 数值解

传统概率模型校验方法需要构建整个概率模型的表征，并由此推导出一个或多个有关将要分析模型的性能结果的数值问题。该方法的第一步往往只需要分析概率模型的基本图（如基于可达性的技术），而其余方法（通常表示在效率方面存在的瓶颈）需要进行数值计算。通常，需要求解线性方程系统问题（DTMC）或线性优化问题（MDP）。尽管这些问题都可出于效率原因通过*直接方法*进行求解（如高斯消元法或单纯形），但经常采用的是*迭代方法*（如高斯-赛德尔法或动态规划法）。只有当收敛达到预期精度时，才会终止每次迭代的收敛解。例如，更多细节请参见文献［32］。

7.3.1.2 近似概率计算

另一种方法是采用离散事件仿真和蒙特卡罗方法相结合来估计与 PCTL 公式关联的概率[17]。这是通过在 DTMC 中生成深度为 k 的随机路径并计算估计 Prob_k

$[\psi]$ 的随机变量值，方程 ψ 的概率满足大多数深度为 k 的路径。具体而言，执行上述过程的算法将 DTMCx 的简洁表示、公式、正整数 k 和两个参数 ε 与 δ 作为输入，产生值 $A(x,\ \varepsilon,\ \delta)$。这是一种*完全随机近似多项式方案*，意味着结果满足

$$\text{Prob}[\ |A(x,\varepsilon,\delta)-\mu(x)|\leqslant\varepsilon]\geqslant 1-\delta$$

式中，$\mu(x)$ 为 $\text{Prob}_k[\psi]$ 的准确值；ε 为近似参数；δ 为*信任参数*。

该算法必须通过 DTMC 生成 $O(\varepsilon^{-2},\ \log\ (\delta^{-1}))$ 路径。该方法的最大优势在于，由于随机路径只能由 DTMC 的简洁表示生成，因此可避免构建模型及其状态空间（这可能成本很高），从而只占用非常少的内存。一种基于验收采样和假设检验[37]的相关技术也需要进行对离散事件仿真的随机采样，但这只是为了检查满足公式的概率是否在给定边界条件内。

7.3.2 概率模型校验工具

PRISM[18,29]是一种开源的概率模型校验工具，最初由伯明翰大学提出，目前由牛津大学开发。该工具可支持 DTMC、CTMC 和 MDP 的分析。采用 PRISM 建模语言对模型进行描述，这是一种相对简单的反应模块形式化为基础的基于状态的语言[2]，同时，以一种集成 LTL、PCT、连续随机逻辑（CSL，对于 CTMC 的一种 PCTL 形式）以及一些自定义扩展的逻辑来规范性能。PRISM 还可支持成本和回报的建模和分析。PRISM 的一个关键特性是采用基于二元决策图（BDD）的数据结构的符号实现技术。这些可允许通过利用从高层描述推导的结构和规律来对超大概率模型进行紧凑表示和有效处理。这种技术已成功应用到状态多达 10^{10} 模型的概率验证（例如，参见文献［11，24］）。

APMC（近似概率模型校验器）[3,17]是一种由巴黎第七大学和巴黎南大学（巴黎第十一大学）合作开发的分布式模型校验器。该工具采用一种有效的蒙特卡罗法来近似满足如前面所述的 DTMC 的 PCTL 性能概率。最新版本的 APMC 还能验证 CTMC，但并未应用于本书所述的工作。该工具包括两大部分。第一部分是一个能够为 PRISM 语言描述的 DTMC 及其性能产生 ad hoc 校验（包括路径生成器和校验器）的编译器。APMC 可实现不同策略以生成与反应模块同步的程序代码：最有效的程序代码称为“编译时间同步”，即预先计算所有组合规则，从而建立同步的简洁模型表示。

第二部分是需要 ad hoc 验证器和一组可用计算资源的配置器，用于在这些计算机上配置验证器并采集模型中公式的近似满足概率结果。APMC 实现了模型校验的一个大规模（但自然）分布方法。根据树状拓扑结构进行配置以实现高效且可扩展的配置。例如，这样可提供一种对数延迟以聚合从所有节点到根节点的结果。树状拓扑的一个缺点是系统可能过度生成和验证某些采样，但可确保在

系统中不存在争议点。关于 APMC 实现的更多信息可参见文献［14］。

7.3.2.1　其他工具

目前，存在一些其他的 MDP 模型校验工具。LiQuor[8] 对利用称为 PROBMELA 的表示语言进行 MDP 建模的 LTL 进行显式状态模型校验，该语言是对 SPIN 中的 PROMELA 语言进行概率扩展。RAPTURE[19] 采用一种迭代抽象细分技术来验证 PCTL 性能的子集。ProbDiVinE[4] 支持并行和分布式的线性时态逻辑模型校验。同时，还有很多支持 DTMC 和 CTMC 概率模型校验的其他工具，其中包括 MRMC[21]、PEPA 插件程序[34]、CASPA[30]、APNN 工具箱[7] 和 Ymer[36]。

7.4　案例分析：CSMA/CD

本节介绍了一个利用 PTA 和概率模型校验进行分析的随机通信协议典型案例分析。在此采用了 CSMA/CD 协议，这是 IEEE 802.3 国际标准（以太网网络通信协议）的一个基本组成部分。

该协议已通过各种方法进行了广泛研究，其中包括仿真[35]、分析方法[13,20] 和实时模型校验[38]。文献［10，28］中进行了在协议概率方面的首次准确形式化分析，其中系统是通过 PTA 建模，并且协议概率行为的正确性是通过分析可达性和时间有界可达性的最小概率和最大概率来进行验证的。

文献［10，28］这两篇论文分别采用不同方法来解决概率（实时）模型校验中通常出现的状态空间爆炸问题。在文献［10］中，时间离散（采用 7.2 节中所述的数字时钟方法），并利用 PRISM[18,29] 和 APMC[3,17] 来分析所得模型。在文献［28］中，PTA 模型是通过在 PRISM 原型扩展上实现的经典时间自动机的模型校验方法进行概率扩展来分析的[16]。

7.4.1　协议

CSMA/CD 是一种分布式网络仲裁协议，其中在此称为站点的多重网络接口卡（NIC）可在称为载体或总线的单个通道上进行通信。所有站点都可向网络发送消息（*多路访问*），并且每个站点都可检测总线是空闲还是正在传送来自另一站点的消息（*载波监听*）。当一个站点感知到总线忙时，则将在传输自身消息前进行等待。然而由于网络间信号延迟传播，站点将可能同时传输消息。在发生这种情况时，将会产生冲突，所有消息均丢失且站点接收到一个乱码信号。由此，站点就可观测到消息已丢失（*冲突检测*）。根据 CSMA/CD，这种情况发生后，在试图重新传输消息之前自主选择随机延迟（称为*补偿时间*），站点可重新安排自身消息传输。已经发生碰撞的数量以指数形式增加时，该随机补偿时间可在冲突发生次数指数形式增大的整个时间间隔内均匀选择。在此，仅着重考虑某一时

刻只能传输一个消息的半双工协议。

7.4.2 PTA 模型

在此采用的 CSMA/CD 协议模型是文献［10，28，38］中的组合模型，并考虑具有两个站点的情况。PTA 模型包括表示两个站点和总线的并行操作的 3 个组件。该模型具有两个参数，即 σ 和 λ，分别表示数据沿总线传播的所需时间和发送一个完整消息的所需时间。

站点 i（其中，$i=1$，2）的 PTA 模型如图 7.2 所示。每个站点都具有一个时钟 x_i。首先，以 INIT 状态开始并试图达到消息已正确发送到其他站点的 DONE 状态。当站点要发送一条消息时（在一定延迟后），将移动到 TRANSMIT 状态。如果在时间 λ 内没有检测到冲突，那么消息将被正确传输并且站点移动到 DONE 状态。否则，若在 σ 时间单位内检测到一个冲突（标记为 cd_i），则站点移动到 COLLIDE 状态。然后产生一个随机等待延迟并等待相应的时间。当等待时间结束时，有两种选择：如果总线空闲，站点将尝试重新发送该消息。如果总线忙，站点增加冲突计数器的值 bc（直到最大值 K），并选择另一随机延迟。

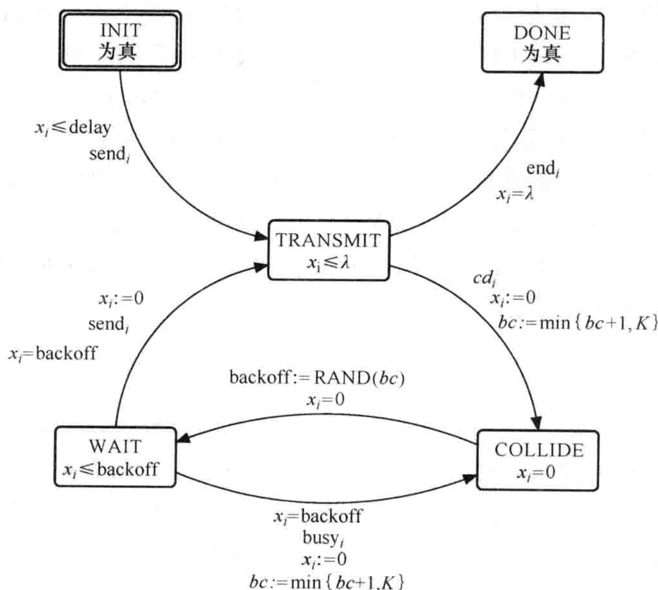


图 7.2 一个站点的概率时间自动机模型

连接两个站点的总线的 PTA 模型如图 7.3 所示。在某个站点准备发送消息之前，总线从 INIT 状态开始移动到 TRANSMIT 状态。如果没有任何发生，站点将完成本次传输（标记为 end$_i$）。在 TRANSMIT 状态时，第 2 个站点只能在表示

总线上传播时间的时间 σ 之前发送消息，此后站点感知到总线忙并不进行发送。总线利用时钟 y_1 和 y_2 来实现上述过程。如果在时间 σ 内发生两个发送行为，则总线移动到 COLLIDE 状态。当接收到其他站点的消息时，两个站点均会检测到冲突，并停止发送，总线将返回到 INIT 状态。

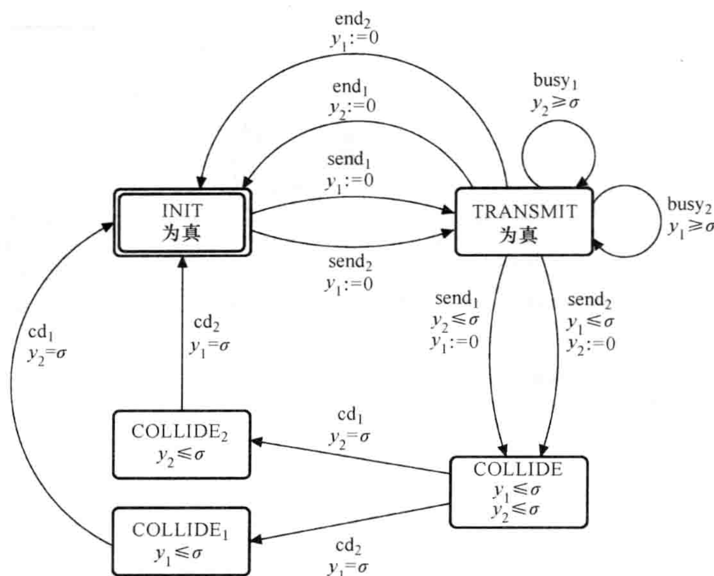


图 7.3　总线的概率时间自动机模型

7.4.3　模型分析

在 7.2 节中已利用数字时钟方法对 PTA 模型进行了分析。通过两个工具 PRISM 和 APMC 对所得 MDP 的概率模型进行校验。对于 CSMA/CD 的模型参数 σ 和 λ，分别取 2 和 65。假设站点初始发送消息（延迟）之前的延迟选择在 PRISM 分析时是不确定的（为 0、1 或 2），以确保考虑到所有可能性。而在采用 APMC 时这是不可能的，因此是随机选择的。

在具有 1GB RAM 且运行 Red Hat Linux 系统的 2.6GHz Pentium IV 便携式计算机上进行 PRISM 实验。当连续迭代产生的解向量元素之间最大相对误差小于 10^{-6}时，迭代数值计算终止。

APMC 实验是在由具有 1GB RAM 运行 NetBSD 系统的 500Athlon3000 + 工作站和具有 512MB RAM 且大多运行 Debian Linux 系统的 20 个 3GHz Pentium IV 工作站通过 100MB 以太网连接为一个异构网格中进行。在计算机和先进技术学院（EPITA）的学生和工作人员使用时，APMC 作为一个“循环窃取者”来执行，

即在工作站的后台运行，因此验证成本的时间统计不精确。然而在网络平均负载较低期间，所有实验都在2天内完成。在所有情况下，除非特殊说明，均采用最有效的APMC策略“编译时间同步”，近似参数 ε 和信任参数 δ 分别为定值 $\varepsilon=10^{-2}$ 和 $\delta=10^{-10}$。

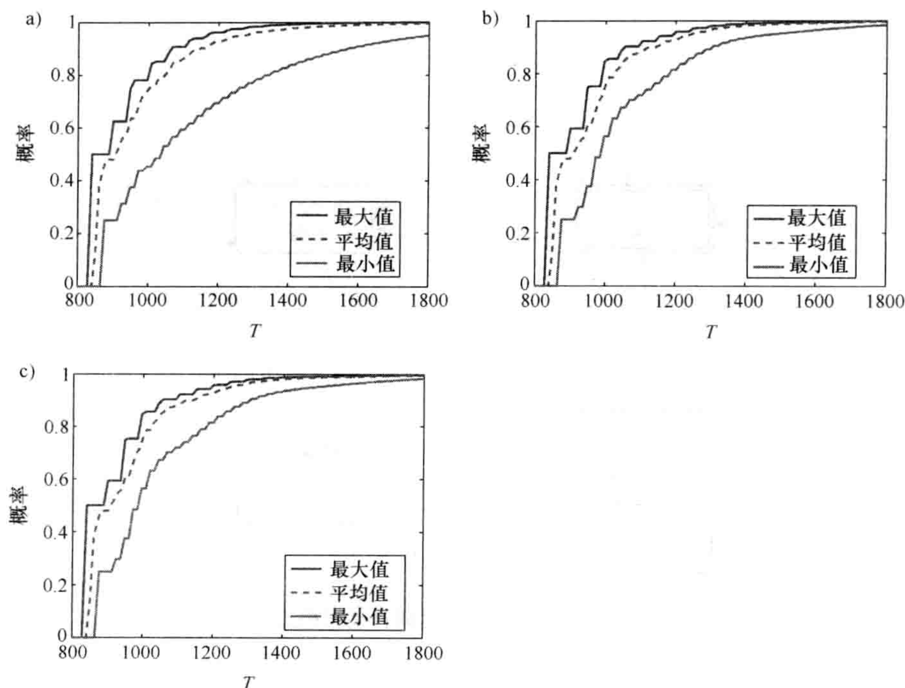


图7.4 时间 T 内的消息发送概率

a) $K=1$ b) $K=5$ c) $K=10$

7.4.3.1 消息传输概率

考虑的首要性能是在特定时间点 T 成功发送一个消息的概率。在图7.4中，给出了最大补偿约束 K（1、5和10，指定标准值为10）和 T 范围内3个值的计算结果。在此给出通过PRISM计算的所有不确定选择上的*最大*传输概率和*最小*传输概率，以及通过APMC计算的假设初始延迟上均匀选择的*平均*传输概率值。

结果表明，采用较大的 K 值可增大时间 T 内发送消息的最小、最大和平均概率。由于是通过站点初始冲突的方式来构建模型，因此在任何限定时间 T 内概率为0，即不允许站点补偿和发送消息。值得注意的是，图中的每个图形（最小和最大）中均包含离散跳跃，这些对应于延迟离散集合上实际的补偿延迟概率选择。平均概率情况下的图形更加平滑且正如预期位于最小和最大值的边界内。

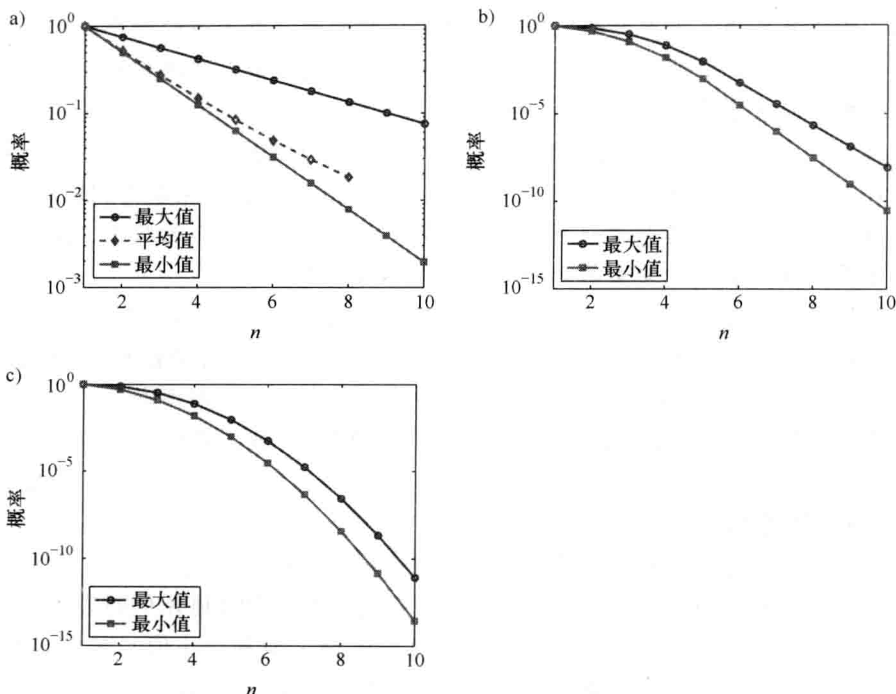


图 7.5　消息发送前站点冲突 n 次的概率

a) $K=1$　b) $K=5$　c) $K=10$

7.4.3.2　冲突概率

接下来，考虑成功发送单个消息之前发生 n 次冲突的概率。对于最大补偿限制 K 的 3 个值以及在 1 ~ 10 的 n 的结果如图 7.5 所示。同理，给出通过 PRISM 计算的最小概率和最大概率。在可能的情况下，还给出通过 APMC 计算的平均概率值。然而，在大多数情况下，由于这些概率值较小，导致计算不切实际。注意到所需样本数为期望近似值逆的二次方程。对于图中对应于 $K=1$ 的一组结果，需采用 10^{-3}的精度，这需要大约 2×10^{-9}个样本。由图 7.5 中的其他图可知，K 值越大，则所需的精度越高，由此所需的样本数不可行。

值得注意的是，由于是以站点初始总是冲突且当 $K=1$、5 和 10 时，发生一次冲突（$n=1$）的最小概率和最大概率为 1 的方式来构建模型。在所有 3 种情况下，结果同样适用于 $n=2$ 的情况，正如 $K=5$ 或 10 时 $n\leqslant6$ 的情况。由于站点的补偿计数器无法达到 $n+1$，直到站点冲突达到 $n+1$ 次，因此这是所期望的。以 $n=1$ 时的情况为例，可阐述产生这些结果的原因如下：

- 最小概率等于 0.5 所对应的是站点在同一时间首次尝试发送消息（没有延迟传播的约束）的情况。在该情形下，站点将在同一时间检测到第 1 次冲

突，因此发生第2次冲突的概率是站点选择相同时隙数等待的概率。因此由于这是等待1或2个时隙数之间的随机选择，站点发生第2次冲突的概率为0.5。

- 最大概率等于0.75所对应的是传播延迟等于σ且站点初始发送其消息之差也为σ时间单位时的情况。在这种情况下，试图传输消息的第2个站点将比其他站点提前σ时间单位检测到该冲突，因此每个站点等待相同时隙数，或在第2个站点决定等待一个时隙时而第1个站点等待两个时隙下，站点将会发生第2次冲突，即发生第2次冲突的概率为0.75。

一般而言，随着n的增大，且当$n \leqslant K+1$时，由于站点在每次冲突时所选择的补偿范围以指数形式增大，因此可认为站点发生n次冲突的概率会迅速减小。然而，一旦$n>K+1$，站点的补偿计数器在冲突发生后不再增加（已达到最大值K），因此站点选择"补偿"的范围保持相同。由此，发生冲突的机会不再急剧下降。

7.4.3.3　预期成本

最后，考虑通过分析考虑成本来增强概率模型的两个性能。首先，假定对应于冲突的每次模型迁移的成本为1，并计算不同最大补偿限制K值时发生冲突的预期个数。这些结果（同样为所有不确定性解的最大值和最小值）如图7.6a所示。其次，假定表示一个离散时间步长的每次模型迁移的成本为1，然后计算完成消息发送所需的预期时间。这些结果可如图7.6b所示。如上所述，原则上，也可能采用离散事件模拟和蒙特卡洛法来产生这些性能的平均概率值。但是，在APMC中尚不支持这些技术，因此在此并不包括这些结果。

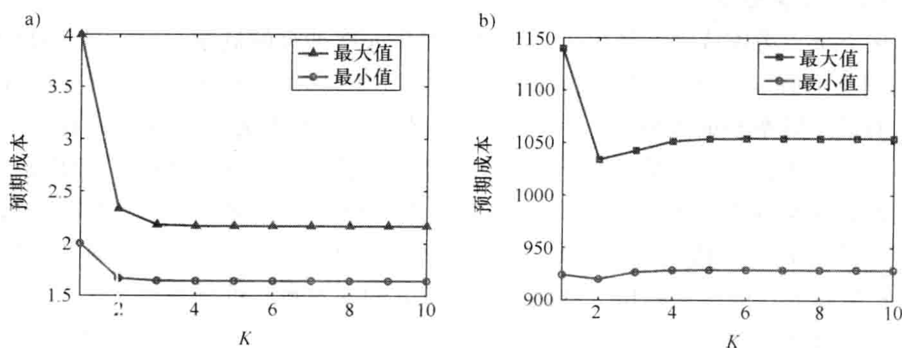


图7.6　发送一个消息之前预期成本的结果

a）成本：冲突个数　b）成本：运行时间

由图可知，随着K值增大，预期冲突个数和预期执行时间最初都减少。这是由于随着K的递增，当试图重新发送时站点发生冲突的机会减小，且消息发送得更快。对于预期的冲突个数，一旦K大于2，结果几乎不变。这是因为大于

3 次冲突的概率很小（见图 7.5）。对于图中预期执行时间的 K 值较大时，也存在类似情况。但值得注意的是，在 $K=2\sim4$ 预期执行时间增大，且 $K=2$ 时预期执行最少。这是因为，尽管发生冲突的次数为 3 或 4 的概率很低，但这些情况下补偿所需的大量时间会增大对预期时间的影响。

7.5　讨论和小结

本章阐述了形式化验证法对通信协议分析的适用性，特别是采用 PTA 和概率模型校验，以及数值解方法或基于蒙特卡罗模拟和抽样的近似技术。同时，还对这两种方法进行对比讨论。

首先讨论每个模型可执行的分析类型。数值解方法的主要优点在于这是基于完整模型的，并通过状态空间穷举搜索来构建。这意味着可能会推导出时态逻辑性能查询的准确解（事实上，正如上述讨论，数值迭代求解方法通常是构建一个期望精度的近似值，而总能得到确切方法）。此外，这种穷举方法可能会计算出最好结果和最坏结果，例如，对模型的所有可能初始配置或不确定性的所有可能解（例如，表示非特定模型参数下进程或多元值之间的并发调度）。

相比之下，基于采样的技术本质上是近似的，且结果表示一种平均行为的概念。然而，该方法适用于一系列更广泛的模型和性能。与只有简单求解算法可用的数值解不同，采样算法可应用于任何一个精确执行随机仿真的模型。同理，通过模型的有限路径可计算的任何性能都可以这种方式进行分析。

进行对比的另一个重要方面是这两种方法的相对效率。显然，改进后数值解的精度和涵盖范围是有代价的。其中采样技术的一个主要优势是，与确切方法相比，执行所需的内存大小只占其中一部分。这是因为可在一个简洁表示的模型上执行采样技术（如某些高级建模形式化描述），以及典型的仿真过程只需在任何时间仅存储一个模型状态。这就使得采样技术可应用于比模型穷举搜索和构建不易实现的数值解更大的模型上。例如，在前面所述的 CSMA/CD 案例分析中，可考虑具有两个站点以上的更大模型配置。

第 2 个优势是，采样技术更容易实现并行或分布式设置，当然这可极大地减少运行时间。这是因为每个个体采样的计算是独立的，而与必须进行大量进程间通信的数值解并行化不同。事实上，这种性能优势是相当重要的，因为正如之前所述，获得准确的结果可能需要生成超大数量的样本。特别是，当计算的实际值非常小时，这是无法避免的。

总的来说，显然根据数值解穷举搜索模型的概率模型校验法和基于离散事件模拟和采样的近似方法都各自具有优点、缺点和平衡。因此，一种大概率系统（如通信协议）的成功分析很可能需要充分利用这两种类型的技术。

致谢

当开展这项工作时，作者得到了法国项目 RNTL“Averroes”（Marie Duflot 和 Claudine Picaronny）、FORWARD 和 EPSRC 项目 GR/S11107 和 GR/S46727（Marta Kwiatkowska、GethinNorman 和 David Parker）以及 MIIJR- FIRB Perf（Jeremy Sproston）的资助。同时，作者还感谢出资人 Laurent Fribourg、Thomas Hérault、Richard Lassaigne、Frédéric Magniette 和 Stéphane Messika，以及 EPITA 使用其计算设备。

参考文献

1. R. Alur and D. L. Dill. A theory of timed automata. *Theoretical Computer Science*, 126(2):183–235, 1994.
2. R. Alur and T. Henzinger. Reactive modules. *Formal Methods in System Design*, 15(1):7–48, 1999.
3. APMC web site. Available at: http://sylvain.berbiqui.org/apmc
4. J. Barnat, L. Brim, I. Cerna, M. Ceska, and J. Tumova. ProbDiVinE-MC: Multi-core LTL model checker for probabilistic systems. In *Proceedings of the 5th International Conference on Quantitative Evaluation of Systems (QEST'08)*, pp. 77–78, IEEE CS Press, 2008.
5. J. Berendsen, D. Jansen, and J.-P. Katoen. Probably on time and within budget—On reachability in priced probabilistic timed automata. In *Proceedings of the 3rd International Conference on Quantitative Evaluation of Systems (QEST'06)*, pp. 311–322, IEEE CS Press, 2006.
6. A. Bianco and L. de Alfaro. Model checking of probabilistic and nondeterministic systems. In P. Thiagarajan, ed. *Proceedings of the 15th International Conference on Foundations of Software Technology and Theoretical Computer Science (FSTTCS'95), Volume 1026 of LNCS*, pp. 499–513, Springer, 1995.
7. P. Buchholz and P. Kemper. Numerical analysis techniques in the APNN toolbox. In Workshop on Formal Methods in Performance Evaluation and Applications, pp. 1–6, 1999.
8. F. Ciesinski and C. Baier. LiQuor: A tool for qualitative and quantitative linear time analysis of reactive systems. In *Proceedings of the 3rd International Conference on Quantitative Evaluation of Systems (QEST'06)*, pp. 131–132, IEEE CS Press, 2006.
9. C. Courcoubetis and M. Yannakakis. The complexity of probabilistic verification. *Journal of the ACM*, 42(4):857–907, 1995.
10. M. Duflot, L. Fribourg, T. Herault, R. Lassaigne, F. Magniette, S. Messika, S. Peyronnet, and C. Picaronny. Probabilistic model checking of the CSMA/CD protocol using PRISM and APMC. In M. Huth, ed. *Proceedings of the 4th Workshop on Automated Verification of Critical Systems (AVoCS'04), Volume 128 of ENTCS*, pp. 195–214, Elsevier Science, 2004.

11. M. Duflot, M. Kwiatkowska, G. Norman, and D. Parker. A formal analysis of bluetooth device discovery. *International Journal on Software Tools for Technology Transfer*, 8(6):621–632, 2006.
12. The Fortuna Model Checker. Available at: http://www.cs.ru.nl/J.Berendsen/fortuna/
13. T. Gonsalves and F. Tobagi. On the performance effects of station location and access protocol parameters in Ethernet networks. *IEEE Transactions on Communications*, 36(4):441–449, 1988.
14. G. Guirado, T. Herault, R. Lassaigne, and S. Peyronnet. Distribution, approximation and probabilistic model checking. In M. Leucker and J. van de Pol, eds. *Proceedings of the 4th International Workshop on Parallel and Distributed Methods in Verification (PDMC'05), Volume 135(2) of ENTCS*, pp. 19–30, Elsevier, 2005.
15. H. Hansson and B. Jonsson. A logic for reasoning about time and probability. *Formal Aspects of Computing*, 6(5):512–535, 1994.
16. T. Henzinger, X. Nicollin, J. Sifakis, and S. Yovine. Symbolic model checking for real-time systems. *Information and Computation*, 111(2):193–244, 1994.
17. T. Herault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate probabilistic model checking. In B. Steffen and G. Levi, eds. *Proceedings of the 5th International Conference on Verification, Model Checking and Abstract Interpretation (VMCAI'04), Volume 2937 of LNCS*, pp. 73–84, 2004.
18. A. Hinton, M. Kwiatkowska, G. Norman, and D. Parker. PRISM: A tool for automatic verification of probabilistic systems. In H. Hermanns and J. Palsberg, eds. *Proceedings of the 12th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS'06), Volume 3920 of LNCS*, pp. 441–444, Springer, 2006.
19. B. Jeannet, P. D'Argenio, and K. Larsen. RAPTURE: A tool for verifying Markov decision processes. In I. Cerna, ed. *Proceedings on Tools Day, Affiliated to 13th International Conference on Concurrency Theory (CONCUR'02), Technical Report FIMU-RS-2002-05*, pp. 84–98, Faculty of Informatics, Masaryk University, 2002.
20. J. Katoen. A semi-Markov model of a home network access protocol. In H. Schwetman, J. Walrand, K. Bagchi, and D. DeGroot, eds. *Proceedings of the International Workshop on Modeling, Analysis, and Simulation on Computer and Telecommunication Systems (MASCOTS'93)*, pp. 293–298, The Society for Computer Simulation, 1993.
21. J. Katoen, E. Hahn, H. Hermanns, D. Jansen, and I. Zapreev. The ins and outs of the probabilistic model checker MRMC. In *Proceedings of the 6th International Conference on Quantitative Evaluation of Systems (QEST'09)*, IEEE CS Press, 2009.
22. M. Kwiatkowska, G. Norman, and D. Parker. Stochastic games for verification of probabilistic timed automata. In J. Ouaknine and F. Vaandrager, eds. *Proceedings of the 7th International Conference on Formal Modeling and Analysis of Timed Systems (FORMATS'09), Volume 5813 of LNCS*, pp. 212–227, Springer, 2009.
23. M. Kwiatkowska, G. Norman, D. Parker, and J. Sproston. Performance analysis of probabilistic timed automata using digital clocks. *Formal Methods in System Design*, 29:33–78, 2006.
24. M. Kwiatkowska, G. Norman, and R. Segala. Automated verification of a randomized distributed consensus protocol using Cadence SMV and PRISM. In G. Berry, H. Comon, and A. Finkel, eds. *Proceedings of the 13th International Conference*

on Computer Aided Verification (CAV'01), Volume 2102 of LNCS, pp. 194–206, Springer, 2001.

25. M. Kwiatkowska, G. Norman, R. Segala, and J. Sproston. Automatic verification of real-time systems with discrete probability distributions. *Theoretical Computer Science*, 286:101–150, 2002.
26. M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of the IEEE 802.11 wireless local area network protocol. In H. Hermanns and R. Segala, eds. *Proceedings of the PAPM/PROBMIV'02, Volume 2399 of LNCS*, pp. 169–187, Springer, 2002.
27. M. Kwiatkowska, G. Norman, and J. Sproston. Probabilistic model checking of deadline properties in the IEEE 1394 FireWire root contention protocol. *Formal Aspects of Computing*, 14(3):295–318, 2003.
28. M. Kwiatkowska, G. Norman, J. Sproston, and F. Wang. Symbolic model checking for probabilistic timed automata. *Information and Computation*, 205(7):1027–1077, 2007.
29. PRISM web site. Available at: http://www.prismmodelchecker.org
30. M. Riedl, J. Schuster, and M. Siegle. Recent extensions to the stochastic process algebra tool CASPA. In *Proceedings of the 5th International Conference on Quantitative Evaluation of Systems (QEST'08)*, pp. 113–114, IEEE CS Press, 2008.
31. A. Roy and K. Gopinath. Improved probabilistic models for 802.11 protocol verification. In K. Etessami and S. Rajamani, eds. *Proceedings of the 17th International Conference on Computer Aided Verification (CAV'05), Volume 3576 of LNCS*, pp. 239–252, Springer, 2005.
32. J. Rutten, M. Kwiatkowska, G. Norman, and D. Parker. *Mathematical Techniques for Analyzing Concurrent and Probabilistic Systems*. P. Panangaden and F. van Breugel, eds. Volume 23 of CRM Monograph Series. AMS, 2004.
33. R. Segala and N. A. Lynch. Probabilistic simulations for probabilistic processes. *Nordic Journal of Computing*, 2(2):250–273, 1995.
34. M. Tribastone. The PEPA plug-in project. In *Proceedings of the 4th International Conference on Quantitative Evaluation of Systems (QEST'07)*, pp. 53–54, IEEE Computer Society, 2007.
35. J. Wang and S. Keshav. Efficient and accurate Ethernet simulation. In *Proceedings of the 24th IEEE Conference on Local Computer Networks*, pp. 182–191, 1999.
36. H. Younes. Ymer: A statistical model checker. In *Proceedings of the 17th International Conference on Computer Aided Verification (CAV'05), Volume 3576 of LNCS*, pp. 429–433, Springer, 2005.
37. H. Younes and R. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In E. Brinksma and K. Larsen, eds. *Proceedings of the 14th International Conference on Computer Aided Verification (CAV'02), Volume 2404 of LNCS*, pp. 223–235, Springer, 2002.
38. S. Yovine. Kronos: A verification tool for real-time systems. *International Journal of Software Tools for Technology Transfer*, 1:123–133, 1997.

第 5 部分

互联网与在线服务

第 8 章　可验证性设计：在线会议系统案例分析

Johannes Neubauer
负责程序设计系统，多特蒙德工业大学，多特蒙德，德国
Tiziana Margaria
负责服务与软件工程，波茨坦大学，波茨坦，德国
Bernhard Steffen
负责程序设计系统，多特蒙德工业大学，多特蒙德，德国

8.1　简介

在线会议系统（OCS）是一种在线投稿和评审服务。这是从 1999 年开始为 Springer Verlag 出版社服务的产品中的一部分[20-21,29]，且随时间的发展，还包括期刊和卷生产准备服务。OCS 作为一种决策支持系统便于核准和拒绝提交的过程。因此，该服务遵循一个对不同应用领域（如会议或期刊）定制的定义明确的工作流。该系统的目的是在合作过程中用于协助不同参与者的高效合作。

自 2009 年以来，OCS 经历了全新的重新设计和重新实现，目标是使之更加灵活和自适应，同时也更适用于验证。图 8.1 给出了如何开展的示意图。首先，描述某些局部模型，如会议的总体评价模式和从用户角度观察的论文生命周期。这些模型可进行模型检查以检查基本特性，包括安全、进展，或简单的可预期因果关系。随后，这些“局部”模型可通过利用复杂的通信和同步原语（如事件处理和进程创建）在企业平台运行来进行半

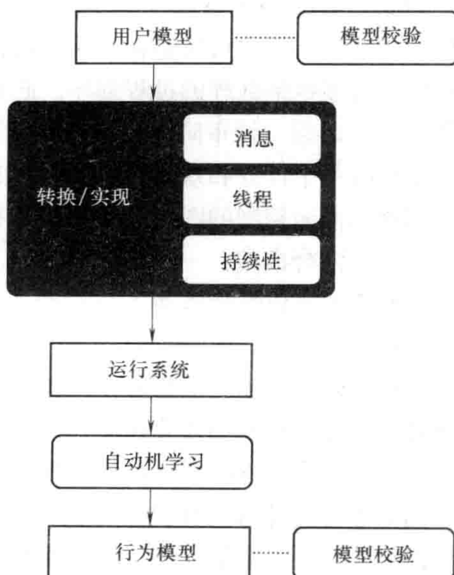


图 8.1　基于模型驱动开发方法的新型 OCS 开发过程模型

自动组合和转换。特别是，这意味着并没有构建 OCS 的任何全局模型。相反，可获得上述转换的充分自由度，然后辅以 8.4 节中描述的自动机学习技术，来回顾推断出全局模型。

本章主要侧重于验证、检验，特别是学习方面，以及将复杂半自动转换过程的处理（图 8.1 中黑色部分）看作一个“黑箱”问题，其中主要是：

- 通过消息的过程同步；
- 通过消息的事件处理；
- 通过（动态创建的）线程的（共享资源）并行处理；
- 进程状态和业务对象的持续性。

具体而言，将描述如何开发（8.2 节和 8.3 节）和验证（8.4 节）OCS 的“局部”模型，以及如何有效地实现这些组合成一个在企业平台上运行的系统（8.5 节）。相应应急行为全局验证的关键是实验性探索通过自动机学习来构建系统行为模型。随后，对该方法可能的规模进行概述（8.5.3 节），并总结相关工作（8.6 节），最后给出小结和展望（8.7 节）。

8.2　用户模型

抽象地看，OCS 的主要目的是适当处理大量独立但通常间接相关的用户交互。因此，可将 OCS 看作一个具有以网页应用程序提供的图形用户界面（GUI）的反应式系统。用户可决定在其执行通常由小工作流自身组成的任务以及多任务时想要处理或拒绝处理的过程顺序。此外，这种交互作用主要取决于评估过程的阶段（提交阶段、评审阶段、讨论阶段等），因此具有明显的控制导向特征。然而，在选择单个任务和众多相关参与者时的较高自由度可排除直接建模由于状态爆炸问题的面向控制的图结构（如自动机、转换系统）[6]。为此决定根据以下条件建立一个混合模型：

- 通过事件和资源共享组织和同步的每个业务实体（如会议、论文）的个体模型集合；
- 控制流类似图结构的个体模型，用于表示整个评估过程中每个所涉及业务对象的个体过程的逐步演变，这包括；
- 在每个状态时系统提供的其他行为声明表示的状态嵌入 ECA 规则㊀。

而对于各种业务实体的个体模型可通过类似模型校验的技术进行验证，如 8.4 节中所述，在此提出另一种处理评估过程整体正确性的方法，即利用自动机

㊀　ECA 规则将用户交互作用建模为一组可并发访问且根据当前事件和相关条件进行选择的规则。

学习从指导性实验的实际实现中学习（推断）全局行为模型（见 8.5 节）。这种方法的优势在于能够抑制复杂设计模型中的所有内部细节以及处理现代企业架构中的复杂通信和同步方法的难度，并且主要侧重于主要问题：用户层的正确性。

图 8.2 将一个会议论文行为的用户模型描述为一个混合图。从语法上，第一个状态是“开始”节点和沿连接线的控制流。边的标签表示触发迁移到下一状态的事件（系统事件或用户交互）。例如，当期限到了时导致系统迁移到具有不同行为的一个不同阶段（状态）所发生的系统事件。

OCS 是一个基于角色的系统[22]，这意味着在一个给定状态下，不同角色可能有不同的作用位。由此，在本模型中可区别这些不同的动作潜能，并通过竖线隔开在一个状态内单独建模。例如，当一个会议激活后（即开始节点之后），会议就处于“摘要提交”状态，作为提交者角色的用户可提交摘要，而其他角色则无法提交。此外，还具有不包含任何内部结构的纯控制流节点，例如，检查条件（如“评阅人分配?”）或向负责人提交问题的系统操作（如“向计算机负责人提交”）。

一个给定状态下的可用行为显然可建模为 ECA 规则。具有丰富控制结构的行为最好采集为控制结构，且作为 ECA 规则的行为不是唯一确定的，这显然是一个设计问题。事实上，如果想要更少状态和更紧凑的图，可以引入比由状态和角色定义的更多前置条件。从语法上说，ECA 规则的前置条件是用在方括号中的一个条件以及一个指示符号（“→”）表示。例如，“[docavail]→download-doc”意味着文档只有在保存后（即上传后）才能下载。后置条件也是类似定义。一个状态内的所有行为都是指 ECA 规则。如果这些行为具有更复杂的条件，则不在用户模型中建模而是在 ECA 规则本身中建模。例如，上传文档可能只限于“PDF”格式。

采用层次结构可实现更简单、结构更好的模型。例如，在图 8.2 中的“评审”阶段，评阅人的 ECA 规则具有黑体字“评审”的行为。这形象地表明该规则下具有子模型，如图 8.3 所示。子模型表明了如何在评审过程中使用子评阅。通过采用层次化结构可大大简化图 8.2 所示的用户模型，此外，还表明子评阅是一个独立行为。事实上，一些会议根本不允许子评阅。

在图 8.2 和图 8.3 中，用户模型描述了一篇论文的工作流。因此，此处的“提交者”和“评阅人”角色与当前所考虑的论文有关，而“PC 成员”和“PC 主席”角色的行为更具有全局性，因为其与正在进行会议的会议服务有关。为具有更好的可读性，图中仅显示了所考虑论文的一个评阅人以及一个相关的子评阅人。

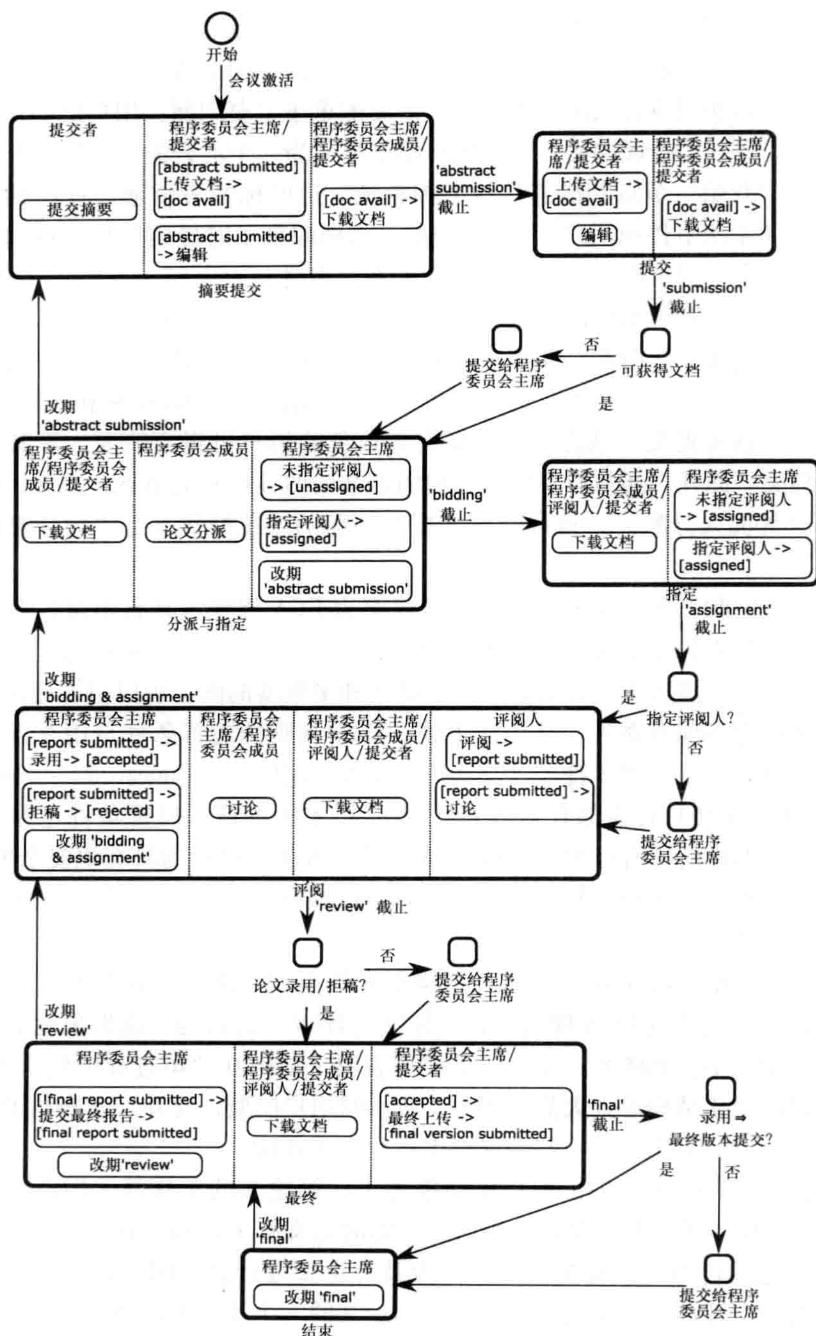


图 8.2 新型 OCS 中一篇论文行为的混合用户模型

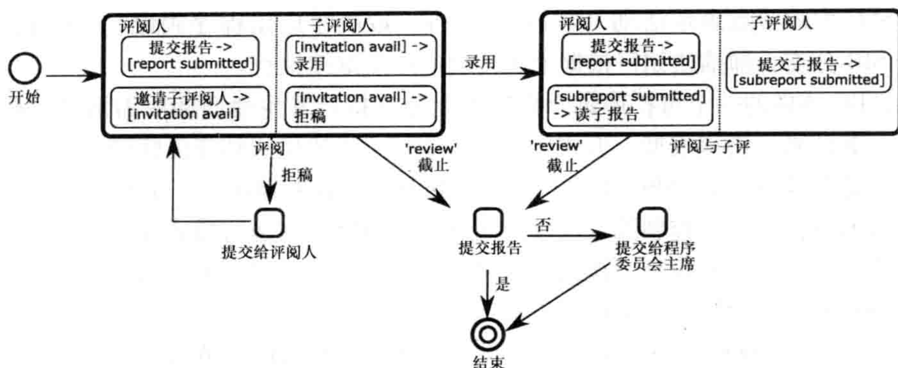


图 8.3　评阅的工作流过程（是指图 8.2 中黑体标记的 ECA 规则"评阅"）

这个建模方式易于模块化定义各种会议和论文的工作流：不同会议的工作流可在独立的用户模型中定义并共存于同一 OCS 系统中。用户模型中每个状态的 ECA 规则集合还可解释为 OCS 中的当前可用功能。在此，利用该信息来动态构建 web 应用程序的 GUI。在运行时，系统查询 ECA 规则评估为真的条件，并对用户 GUI 给出相应的行为集。这个动态建立 GUI 的方式与会议服务的具体工作流无关，因此适用于任何可能出现的（会议）服务。

刚刚讨论的图本身就是形式化的模型，是在 jABC（8.3 节）中建模并如 8.4 节所述进行模型校验。

8.3　模型与框架

OCS 模型在 jABC[33,48] 中实现，这是一种面向服务设计和允许用户通过将可重用构建块组合成相对形式化且易于阅读和构建的（流）图结构进行服务和应用开发的框架。类似于通信中的术语[45]，这些构建块称为*服务无关构建块*（SIB），并且在面向服务的计算范式[28,35]和 *One Thing* 法[46]的启发下，将一种文献［32］中基于模型的轻量级协调方法的进化算法应用于服务。SIB 是可参数化的，从而使得其行为可根据当前应用背景而自适应。此外，每个 SIB 有一个或多个指向 SIB 的继承构建块的输出支路。在运行时确定采用 SIB 的哪个分支。

在一个大型 SIB 库的基础上，用户对期望系统构建称为服务逻辑图（SLG）的层次化图模型[4]。在一个 SLG 中，一个 SIB 可表示一个单一功能或整个子图（即另一个 SLG），从而可作为一个隐藏更详细进程模型的宏。此功能可提供不仅是组件而且在更大应用程序的整个（子）模型的更高可重用性。

SLG 可从语义上解释为 Kripke Transition Systems（克里普克转换系统，KTS），这是允许节点和边上标记的 Kripke 结构和标记转换系统[38]的一种通用形

式。SLG 中的节点表示活动（或服务/组件，取决于应用程序领域）。边直接对应于 SIB 分支，即描述如何根据上次活动的结果继续执行。

jABC 插件的一个可扩展集合提供可充分支持开发生命周期所需活动的附加功能，如动画、快速成型、形式化验证、调试、代码生成和变更管理。

跟踪器插件可为 jABC 中的 SLG 提供执行层。跟踪器可将一个 SLG 解释为能够跟踪的可执行有向控制流，这类似于标准调试器中在运行模式或单步模式下通过断点或暂停来中断执行。该跟踪器可应用于整个开发周期来测试从一个初始模型快速仿真到后续阶段更全面模拟的运行时行为。

一旦一个 jABC 应用程序的 SLG 设计完成且所有 SIB 都被执行，图就可以进行代码生成和配置。这是 Genesys 的任务[19]，即对不同目标平台和语言（如 Java、C#、BPEL 和 Android）提供高度专业化代码生成器的一个 jABC 插件。

然而，将一个 SLG 解释为 KTS 是能够形式化验证这些模型并在设计时确保其遵循一组特性的核心功能。为了支持以面向性能的方法来变更管理，这不仅在初始开发阶段非常重要，而且在系统演化时尤为重要。

8.4 模型校验

在整个开发过程中利用模型校验来确保对于个体业务对象能使得模型满足特定性能。从系统的观点来看，这种行为可看作子系统级的验证。在此没有全局设计模型。相反，全局系统行为只能通过实现各种子系统的进程实例间的实际实现的交互作用而产生。正如 8.5 节中所述，全局行为可利用特定的自适应自动机学习技术对模型外推进行验证。

对于表示为 SLG 的子系统模型的验证性能可用计算树逻辑（CLT）[6] 进行表示，并利用参数化工具箱和 Hennessy-Milner 逻辑（HML）的钻石运算符（即 < >）进行强化[12]：

$$\Phi ::= AP \mid (\neg \Phi) \mid (\Phi \wedge \Phi) \mid (\Phi \vee \Phi) \mid AG\Phi \mid A(\Phi U \Phi) \mid [a]\Phi \mid \langle a \rangle \Phi \quad (8.1)$$

语义解释如下：

- AP 表示原子命题，特别是 tt 和 ff，常量真和假。
- $\neg$，$\wedge$ 和 $\vee$ 是常见的布尔运算符。
- $AG\Phi$ 意味着“在每个状态的每条路径，Φ 成立。”
- A（$\Phi U \psi$）的意思是“在每条路径上，Φ 成立直到 ψ 为真。ψ 需要在每条路径的（至少）一个状态上成立。在 ψ 为真的状态上，Φ 可能为真或为假。”
- $[a]\ \Phi$：“在所有‘a’的下一节点，Φ 均成立。”
- $\langle a \rangle \Phi$：“在至少‘a’的一个下一节点，Φ 成立。”

CTL 是一种广泛用于表示本例中由自动机建模的系统性能的形式化方法。在本书中，CTL 也由 Siminiceanu 和 Ciardo（见第 5 章）来推理航空电子系统的性能。

图 8.2 所示的用户模型（在 8.2 节中已大致描述）在整个会议服务系统的循环周期中描述了处理论文提交的控制流，该过程由几个阶段组成（如“摘要提交”、“论文提交”、“招标”、“评阅”、“最终稿”），每个阶段都有特定行为。例如，只应在“摘要提交”和“论文提交”阶段允许论文文档的上传，且性能可由本书提出的 CTL 表示：

$$AG([\,'\text{submission}'\text{expires}\,](AG(\neg\ \text{upload doc}))) \tag{8.2}$$

表示在标记为'submission'expires 的边传递后禁止 upload doc 的上述表达式可通过模型校验易于检查描述论文处理的 SLG。令人惊讶的是，这证明性能不成立。这是由于一些具有允许重新安排提交阶段（参见图 8.2 中边重新安排指向“'abstract submission'”阶段的“'abstract submission'”）的（常见）OCS 功能的“功能交互”所致[17,18]。由于并不想丧失该常见功能，因此必须使得性能更宽松以容忍如下列形式的附加行为：

$$A\ G([\,`\text{suhmission}'\ \text{expires}\,](A((\neg\ \text{upload doc})U(\text{Abstract Submission})))) \tag{8.3}$$

8.5　通过自动机学习的应急全局行为验证

本方法用来验证全局系统性能的特点是明确上述概述的子系统“传统”处理和基于学习的全局行为研究之间的关系。正如对于许多实际系统，将子系统的局部模型扩展为 OCS 全局行为的形式化模型（正如在文献［22］中所进行的基于角色的工作）非常繁琐，这是由于会涉及各种通信和同步范式，如事件处理、共享资源，甚至进程创建，以及对于大量运行进程实例的所有行为。为此，采用了一种不同的观点，即主要仅关注全局用户进程以及重要的可见影响。自动机学习（见第 11 章以及文献［16，41］，具有通过在企业平台上运行应用程序的系统实验来推断或外推该用户进程的近似和视图的潜力。根据实验/测试的基础，该方法具有以下优点：

- 不会出现否则不可避免的状态爆炸问题；
- 在具有所有复杂通信和同步机制的实际企业系统上的研究基础；
- 通过着重于特定活动的抽象、抑制细节和投影可对建模模型进行微调；
- 建模模型的精度可以提高系统的生命周期：外推模型和在基于模型的监测中表现的实际系统之间的差异可直接用于细化模型。

所产生的模型可用于生成测试，尤其是回归测试，通过模型校验（见图 8.1 中的过程模型）的验证，以及一些人工检测。接下来，在表明如何利用领域知识来使得该方法适用于更大规模的系统模型，直到实际系统模型之前，将首先学习

全局用户行为的某些抽象视图来阐述该方法。

8.5.1 学习设置

本节介绍了如何将第11章中讨论的学习方法应用于OCS。这特别是需要确定一个允许操控实验并观测系统基本结构的适当字母。由于LearnLib的灵活性[39-41]，可直接推导特定SLG的模型，这从结构上非常类似于Mealy机[30,43]，即由LearnLib特定支持的模型结构[30]。在对于基于学习的系统查询进行测试案例动态定义的基础上，通过对web客户端的OCS应用程序编程接口（API）给出相应的输入字母表。这些测试案例的响应构成输出字母表。如图8.6所示的SLG是一个小型学习模型的示例。其中，具有4个㊀由圆形节点表示的状态和25个㊁由矩形块表示的迁移。这些矩形块的标记来自于输入字母表（见图8.4中随后考虑的初始输入字母表），而从矩形块输出的边上由来自输出字母表的符号进行标记（在此为*default*和*error*）。下一节将对这种建模模型进行直观描述。



图8.4 输入字母：一组用户级的原子行为

这种SLG的表示相当冗长，但具有两个优点：

- 由于矩形节点直接对应于OCS的API调用，因此这些模型在jABC中可直接执行；
- 随着jABC立刻成为一个相应测试环境的影响，还可用于表示学习过程中测试案例的动态创建。

在图8.4中显示的SIB用于测试案例之前，必须完全实例化。例如，登录SIB必须填写用户名/密码，且通常情况下，至少需要包含其中一项正确和一项错误的组合形式来研究所要学习的系统。

学习过程本身也需要一些准备：

- 由于动态创建的测试案例必须是独立的，因此在“崭新”系统中以每

㊀ 参考图8.7，state0 ~ state4。——译者注

㊁ 参考图8.7。——译者注

个测试案例开始是非常重要的。根据在第 11 章中将要描述的重置抽象技术，通过对每个测试案例在 OCS 中都创建一个新的会议来实现。

- 迭代学习过程可产生所谓的假设模型，即与现有观测结果相一致的中间模型。然后通过所谓的等价查询来检查这些假设是否与期望模型匹配。对于 OCS，提供增强的输入符号（参见图 8.5）使得能够更容易地发现假设与系统之间的潜在差异。

接下来，还是根据这些前提条件来相当初步地处理 OCS。

8.5.2　学习行为模型

一旦图 8.4 所示的字母表完全实例化，则自动机学习器可生成如图 8.6 所示的 SLG。这种在实际 OCS 中本身可执行的 SLG 描述了在一个由一组输入符号隐式定义的抽象层上的 OCS 行为。

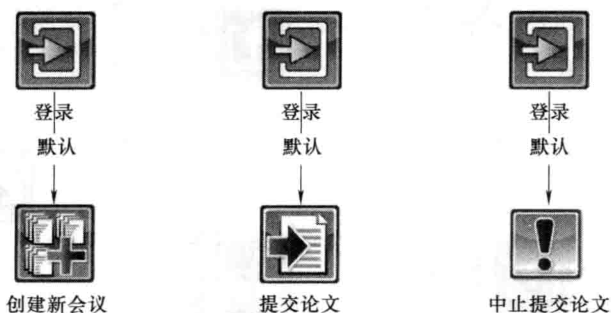


图 8.5　作为字母增强的输入序列

在这个 SLG 中，标记为 state <number> 的 SIB 对应于 Mealy 状态机的状态。从这些状态输出的边到达 SIB 对应于 OCS 的输入符号。例如，在 state0，用户可登录或注销。这是因为可以从 state0 到标记有这些行为并得到确认的 SBI 上的另一状态（即 default 的输出边）。另一方面，在这种状态下，用户既不能提交论文也不能创建新的会议，因为这些行为会产生否定裁决（即输出边 error），从而在当前 OCS 中不可行。

图 8.6 中的 SLG 是由等效 oracle 数据库的标准近似获得的，即设计一个 oracle 数据库来检测有差别的运行，这是运行在假设模型与实际系统之间的对称差异。这个简单的 oracle 数据库可简化查找假设自动机中每个状态现在和之后的差异。由于在实际中通常不可能实现等价 oracle 数据库（参见第 11 章），LearnLib 可提供一组不同的近似集合。一个简单扩展到“预先一步”版本可利用基本输入符号队列来增加输入字母表，正如图 8.5 中的 OCS 所示。直接增加这个字母表会导致产生图 8.7 中所示的细化五状态模型。例如，这种由 3 个组合字母符号造成的稍微预先扩展足以对等价查询进行简单近似来检测假设中的一旦注销就无法提交论文，甚至在重新登录后又可以提交论文。

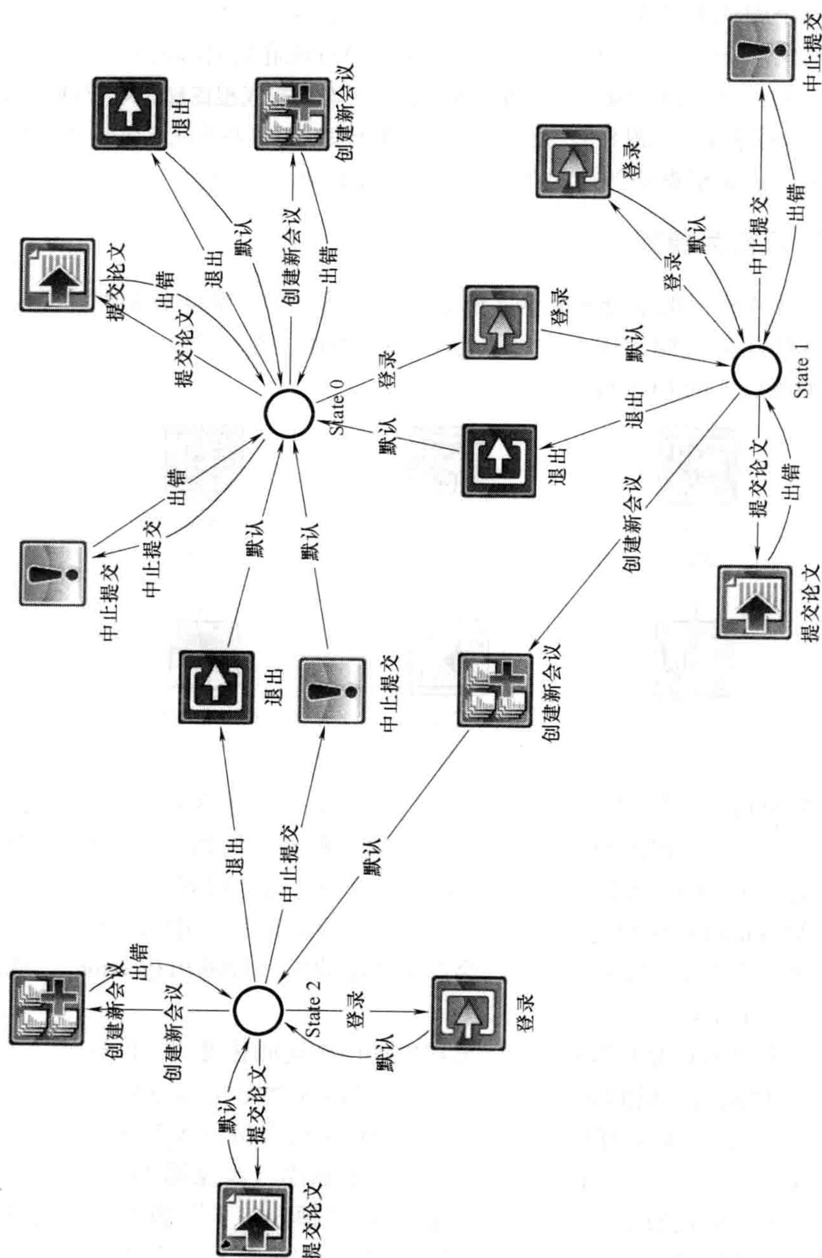


图8.6 只具有原子字母的Mesly学习状态机

8.5.3 节阐述了如何增强学习以获得真正的实际行为模型。事实上，利用 LearnLib，已经能够学习具有上万个状态的模型[40]。在实际应用中，一个主要的障碍是如何实现一个足够等价的 oracle 数据库，即能够快速检测判别运行的 oracle[13]。实际上，在此人工指导可能更易于处理。这是由可执行假设模型的 SLG 所支持：即对于矩形 SIB 是自动执行的，且每当执行一个（圆形）状态 SIB，都会弹出一个对话框，为用户提供下一个输入符号的选择，由此可允许用户引导搜索判别运行。

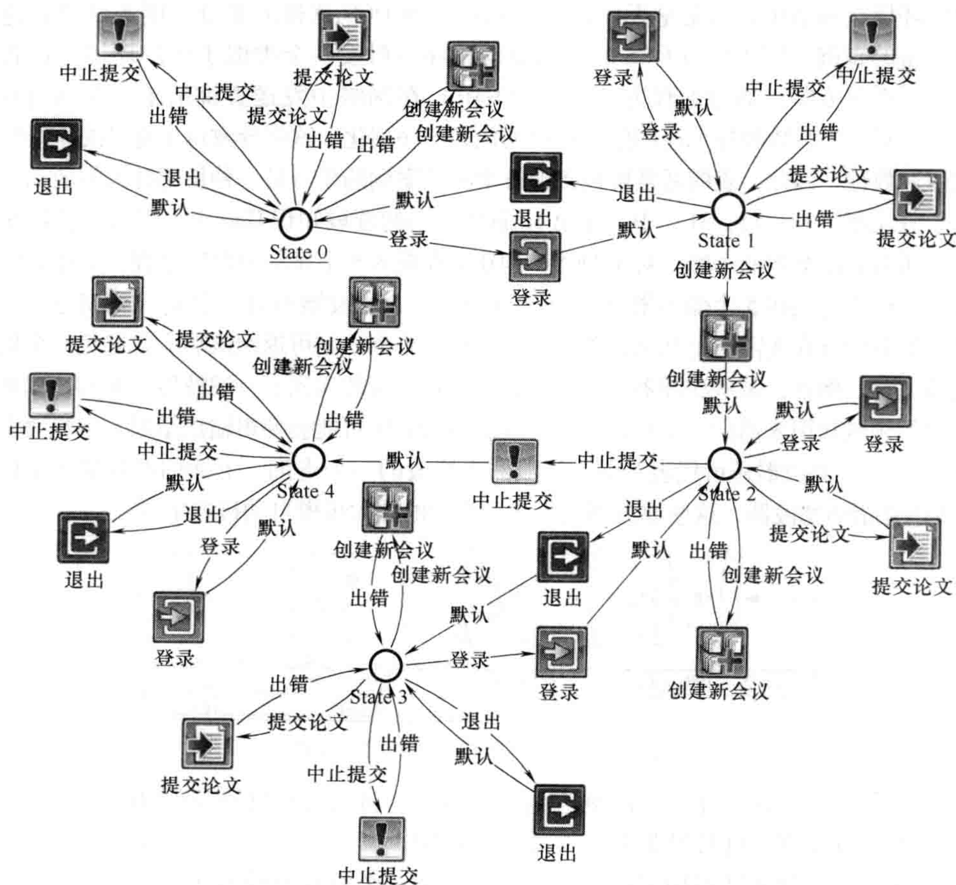


图 8.7　具有符号序列的 Mesly 学习状态机（图 8.5）

更复杂的是一个基于模型校验的搜索判别运行的可能性。由于 SLG 可以很容易地进行模型检查，因此可制定假设对 OCS 成立的时态公式。如果该公式对假设模型不成立，则可作为在 OCS 上进行测试的一个具体候选。如果测试成功，则可以确定一个判别运行。事实上，通过一个表明在中止提交阶段之前总是可以最终提交论文的公式，可实现将图 8.6 中的模型细化为图 8.7 中的模型。

8.5.3 便于领域知识的自动机学习

学习需要在所要学习的系统上进行大量测试执行。即使没有进行任何假设和实际系统一致性检查的测试，也需要至少 kn^2 次测试运行（在学习相关文献中称为“会员查询”），其中 n 为状态个数，k 为输入个数。一个具有 30 个状态和 12 个输入的模型需要大于 10000 次的测试。在一个只对执行某一查询进行相关局部计算的模拟环境或场景中，这是毫无问题的。但对于像 OCS 这样的系统，则无法进行这种大量的查询。用于学习 OCS 的测试驱动程序可触发一个类似于生产环境中设置的实际系统安装。通过远程方法调用（RMI）在网络中发送查询请求。在该过程中，应对每个参数和每个返回值进行序列化和反序列化。这将导致每个符号的执行时间多达数秒。因此，表明通常可包含上万个测试案例的学习是一种长期持久的活动。

在文献［14，15，31］中，在第 11 章中介绍的领域知识基础上，引入滤波器来减少所需测试案例的个数。对于 OCS，构建了在图 8.8 中显示的设置过程，其中，实际的测试执行是在 5 个滤波器条件下进行的。一个滤波器可对一个输入查询进行评估。如果能够在无需执行新测试条件下应答整个查询，则可返回该响应并避免冗余的实验工作。例如，某些滤波器可修改查询，只将前缀传递到下一滤波器，并利用增加后缀的领域知识来返回一个级联响应。滤波器的顺序可能会使得输出结果发生显著变化[31]。一个滤波器还可代表对下一滤波器无修改的一个查询。下面的罗列描述了图 8.8 中介绍的滤波器，这些滤波器用于之后介绍的更大规模自动机的学习：

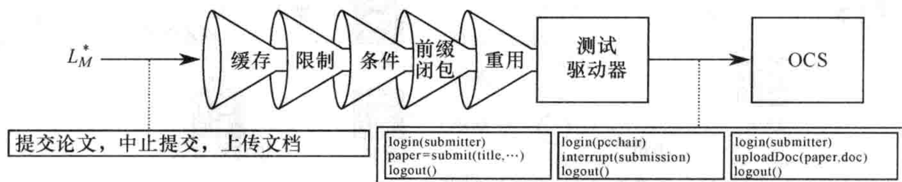


图 8.8 对于 OCS 的过滤器学习设置

缓存。学习算法在某一时刻产生冗余查询。特别是在处理反例期间。缓存滤波器可维持所有的计算结果和直接复制的响应。

限制。限制滤波器用于限制在一次测试中一个符号出现的次数。例如，若滤波器设为 1 而符号出现两次，则滤波器丢弃该符号的后缀并生成输出符号 error。

条件。条件滤波器可截断作为前置条件失去必要符号或禁止前个符号的字节后缀，并对剩余后缀返回输出符号 error。

前缀闭包。OCS 采用一个事务管理器。如果发生错误，则活动事务在不改变系统状态下重新运行。这个前缀闭包滤波器便于实现该性能并可截断失败查询的后缀，然后返回输出符号 error。

重用。重用滤波器通过重用现有的会议服务来省去重置和符号执行。如果要

求一个测试的前缀，则重用滤波器将会执行相应会议服务的后缀并返回已经执行前缀和刚执行后缀的级联输出符号。

图 8.9 描述了图 8.10 中所示关于具有部分转换函数的确定性自动机的相应

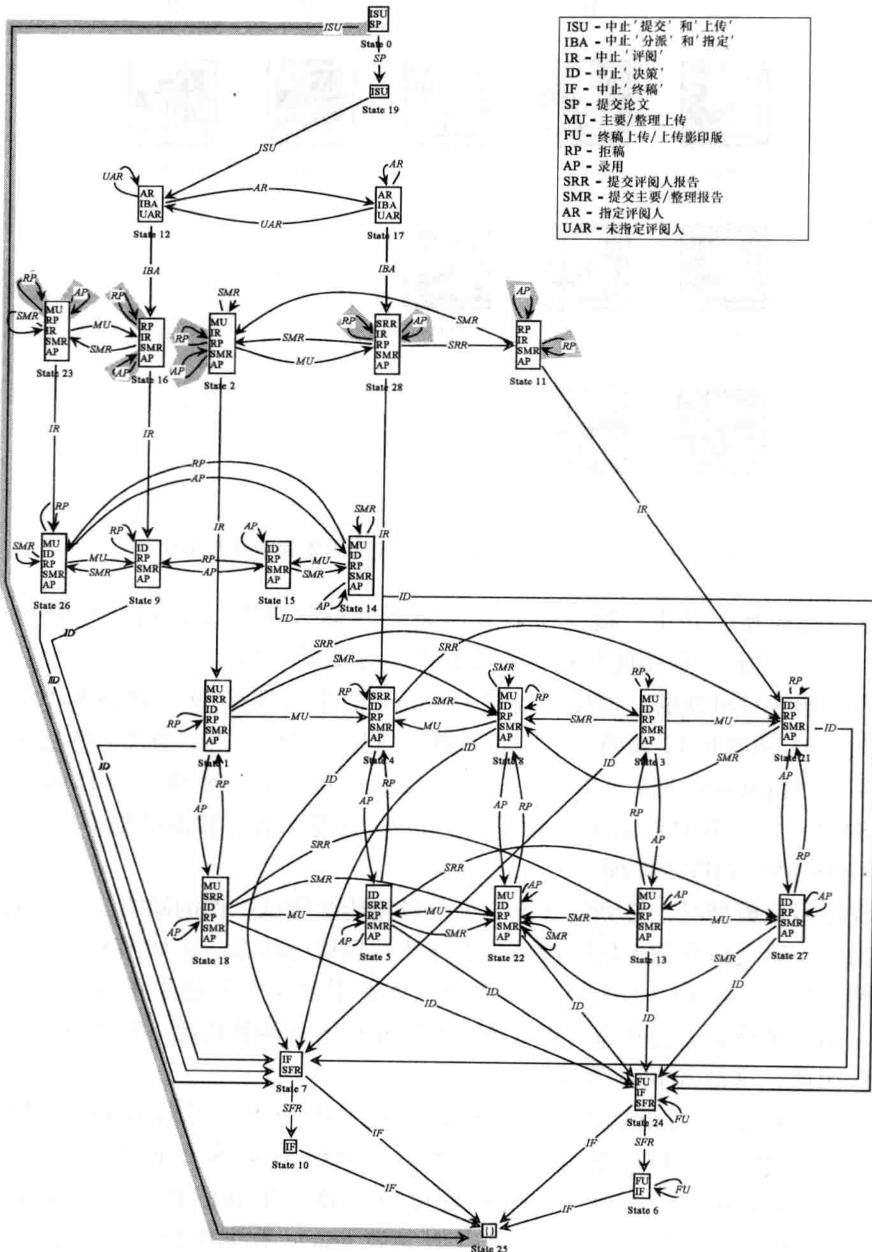


图 8.9　具有 12 个字母符号的 OCS 行为模型

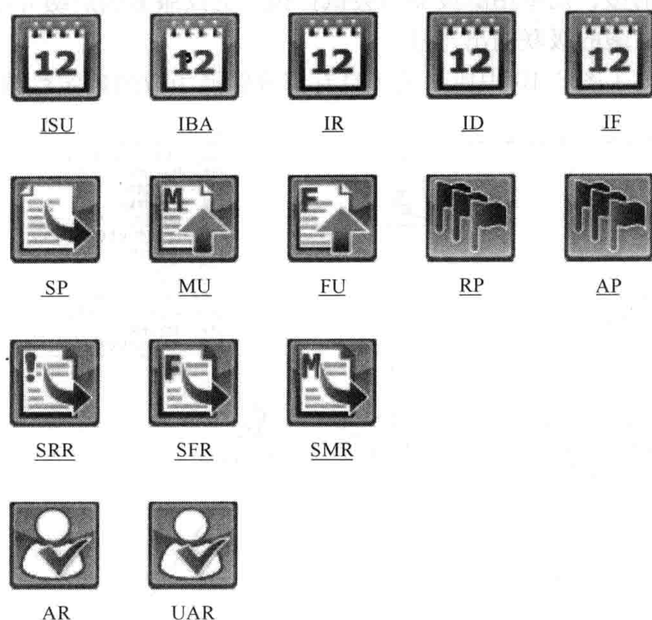


图 8.10 12 个符号的字母表（认证已集成到字母符号中）

字母表学习模型，其中，接受所有状态且缩写输入符号（但仍然是唯一标识的）。由于自动机为前缀闭包且消除所有错误分支后只拥有一个输出符号，因此这种表示形式是可能的。该模型经过 1h58min 的学习具有 29 个状态和 107 个迁移。图 8.11 和图 8.12 表明了不同滤波器的效果。简而言之，该 L_M^* 算法采用了 31 971 个查询和 432 558 个符号。应用滤波器可将测试次数减少到 1518 次，且只有 8418 个符号在被测系统（SUT）下执行。这意味着滤波器抑制了 95.25% 的查询和 98.05% 的符号处理。

在学习该模型时，检测到系统中一个避免传统测试工作的缺陷，即在论文提交之前中止论文提交和上传阶段可立即导致达到最终状态，正如图 8.9 中突出显示的错误轨迹。在 OCS 实现的具体层面，该错误对应于上述中止后的一个死锁状态。由模型终止下的状态所确定的该死锁是由于在两种情况下只能继续执行空字节。因此，这是 Nerode 等效的。

与此同时，学习模型仍是不正确的，这是因为近似等效的 oracle 数据库太小而无法体现所有可观测状态。例如，突出显示的在状态 2、11、16、23、28 下接受和拒绝一篇论文的自反边应导致产生新的状态，这类似于状态 1 和 4 的情况。目前，正在改进近似等效 oracle 数据库以实现该问题的自动求解。

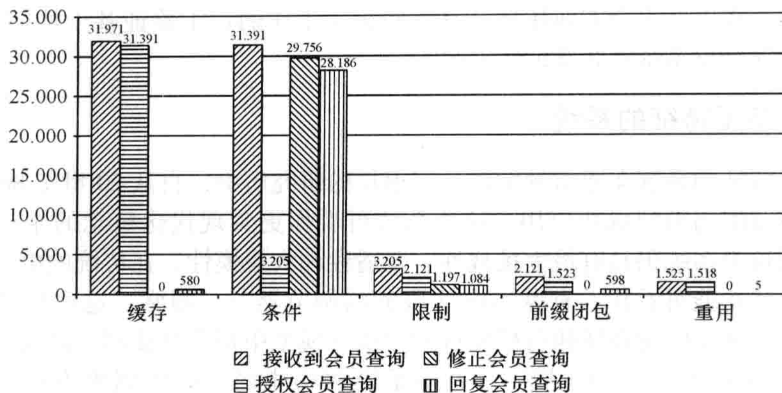


图 8.11　滤波统计（会员查询）

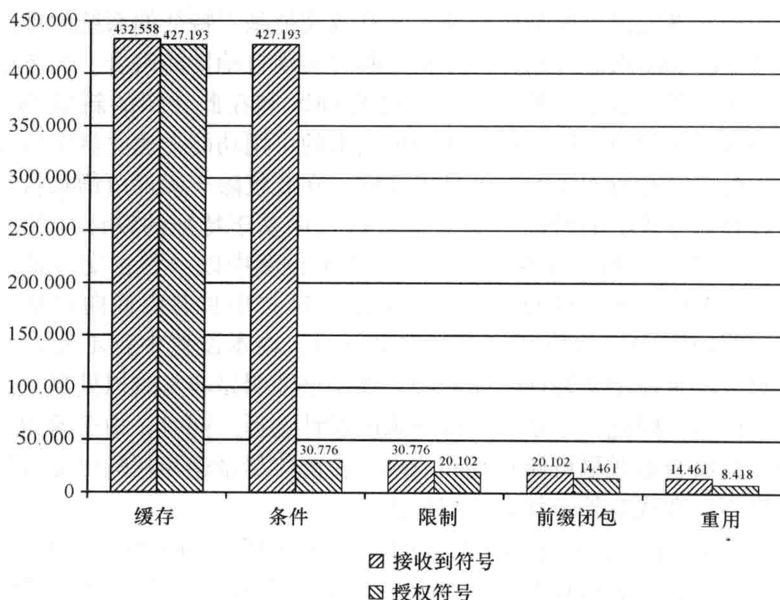


图 8.12　滤波统计（符号）

8.6　相关工作

采用自动机学习的方法来弥补局部特征模型之间及其在具有复杂先进通信和同步设备的企业平台上的组合应急行为之间的差异是一种全新的方法，且据人们所知，目前还没有有关该技术类似应用的文献。

但是，在以下3个方面还是具有一些相关工作的：①验证基于特征的系统；②验证在线会议系统；③验证一般政策。

8.6.1 基于特征的系统

基于特征的系统在通信领域已具有很长的研究历史，自从20世纪80年代末已在全球范围内得到成功应用。这些系统可看作更具现代化概念的生产线的前身，实现这类系统仍具有最大挑战性、普适性和大规模性。在传统的电话服务设置中，这些功能可看作一种基本电话服务的调节器[5]。通常，这些功能是顺序执行的，在所谓的菊花链执行模型或顺序执行模型中每个功能都远离又回归基本服务[44]。在基于web的应用中，基本系统和功能之间的比例更为悬殊：基于web的应用程序具有一个最少的骨干服务且几乎完全由功能构成。这种基于特征的系统的建模和验证已进行了相当长时间的深入研究：例如，在文献［3，26］中提出了一种先进的组合模型校验技术来有效规范基于特征的系统的性能。其目标是能够对功能和需求进行划分，并在校验时实现自动性能组合。为考虑服务所需的复杂演化及其灵活适应性，对于最初的OCS和在此介绍的新型OCS，允许一种特征的多层次组织结构，即在其他更基本的可用功能上建立特定特征。为保证这种结构可控以及所产生的行为易于理解，在此仅限于确保可增加行为的单一功能。通过其他背景下的特征（如文献［11］中基于特征的设计）和类似于面向方面的设计[24]中的特征实现的限制行为在本设置中以一种正交方式实现，即通过需求层的约束。通过机制考虑（如文献［11］中具有由面向对象设计重新编写产生的明确影响）的特征来重新定义行为，在本设置中是不允许的。由于试图定义和分析重新定义特征之间的相互作用已证明是非常难以处理允许修改和抑制的特征模型，因此，这是一种很明确的设计选择。同时，由于希望行为尽可能地组合，且组合形式尽可能直观和清晰，在此应尽量避免任何可能导致影响透明度并产生不一致或意外干扰的建模特点。

与文献［5］相比，在此是从特征的合法利用中来区别特征行为的描述。实际上，行为的限制是表现在不同层面上的，即在需求层（通过时态逻辑约束），并是希望能够利用形式化验证方法自动校验的面向方面的性能描述中的一部分。在8.4节中所述的所有性能都是以这种逻辑表示的需求，同时这些性能也是在文献［1］中确定的作为计算机支持的协同工作平台特点的安全性和一致性的需求实例。这些实例是基于角色的访问控制（RBAC）模型中的具体实例[42]。能够通过模型校验来自动验证上述性能是jABC的一个显著优势，这对于确保所构建的这类应用程序的安全性是十分必要的。

8.6.2 在线会议系统

关于在线会议系统，考虑到许多直接来自于计算机科学家团体，且这些是团

体甚至作者自己提出的验证方法设计在应用和检验方面的一个良好领域，但令人惊讶的是，很少在科学会议上发表有关这些工具的论文。在 Cyberchair[49] 和 Continue[25] 发表有早期的论文，但很少有如何对这些系统进行验证的具体信息。毕竟，此类提交系统应包含整个团体尚未发表的知识产权，因此有理由期待团队获悉如何保护知识产权的极大兴趣。

关于验证，Continue 已作为大量关于建模和验证方面论文的案例分析[9,27,36]。其关心的是利用抽象状态机来建模动态策略[9]，确保浏览器的导航不会破坏实施限制访问的政策[27,36] 和对于政策验证 Margrave 的模型校验[8]。在这项工作中考虑的动态方面只是每个角色面向阶段的正常权利切换，这作为 SLG（见图 8.2）很好地采集，而无需依靠任何额外的和更多的形式化技术。在一些论文中已介绍了 OCS 的形式化验证[21-23]，尤其是角色和权利的处理以及与其他策略描述语言的对比。

8.6.3　政策

通过复杂在线服务访问敏感数据需要一个适当机制来定义、验证，并制定确保服务信任度的政策[7]。在文献［23］中，介绍和比较了基于 XML 的描述语言 XACML[10,37] 和具有约束的 WS-Policy[2] 以及在 OCS 中采用的用于对访问控制建模的基于图的方法。从变更管理和演化的角度，需要一个结构化且灵活的政策模型来处理动态性，特别是在许多用户具有不同角色的系统中处理权利。虽然 XACML 和 WS-Policy 足以描述局部的角色/权利模型，以及充分表示静态策略，但对于 OCS，需要表示面向过程的政策：将局部政策嵌入在一个时态组件中，且在线性时间时态逻辑或本例中的分支时态逻辑中。这是通过基于 jABC 的功能来涵盖，但不会由 XACML 或 WS-Policy 涵盖，这是一个纯粹命题。此外，通过模型校验的政策约束的基于模型的验证是非常重要的，以保证服务控制机制的可靠性。这使得连续一致性检查沿着增量改进和在服务的整个生命周期中演化，这种方式充分支持服务与其策略集之间的配准。目前，这是在 jABC 中提供的，而在 XACML 和 WS-Policy 中尚未提供，因此这仍然无法满足需要。

8.7　小结和展望

在此，引入一种新的混合建模方法，旨在克服直观的用户级建模和在企业环境中运行复杂应用程序之间的差距。本方法的前置条件是对基本的业务实体（如 OCS 中的会议、论文和其他业务对象）按照控制流的类似图的结构进行建模。这些结构表明了业务实体在整个评估过程中的逐步演化，其中包括明确表示每个状态下系统提供的其他行为的状态嵌入式 ECA 规则，并有助于将这些“局

部”模型半自动地整合到一个全局系统中。全局系统通过复杂的通信和同步原语（如事件处理和过程创建）在企业平台上运行。由此，各种业务实体的个体模型可通过模型校验的技术来验证，但由于没有相应的全局模型，因此无法验证全局系统。现已表明如何利用自动机学习通过指导性实验从实际实现中推导全局行为模型来克服该缺点，从而可支持在系统层面上的检验和验证。整个开发都与*增量式形式化*[47]和*连续模型驱动工程*[34]的原则保持一致，在此，主张通过形式化方法支持设计渐进却一致的形式，这不仅应用于最初的设计，而且贯穿于系统的整个生命周期和演化。

OCS 涵盖了许多不同的应用领域。目前应用于会议、期刊和其他形式的提交和评阅过程，但可以很容易地定制以支持其他基于委员会的分布式决策支持实例。支持所有形式需求的复杂性都可通过在建模层面上引入生产线来处理。所有会议、期刊或其他服务可共存于一个 OCS 实例中。这样，资源得以保存，且所有服务都取决于一个一致的用户数据库。

到目前为止，仅学习了 OCS 的一小部分。下一目标是学习更大的部分，这需要在以下方面进行研究：

- 增加所考虑行为的字母表；
- 优化学习过程；
- 进一步利用可用领域知识通过滤波器减少真正执行的会员查询，
- 建立行为模型的视图，以侧重于模型的特定问题，
- 详细说明学习过程的抽象程度，从而得到一个合理的模型（如限制一个会议上的论文数或一篇论文的评阅人数）。

幸运的是，随着 LearnLib 的发展，所提出的学习框架确实在这些方面逐步完善。因此有信心能够很快展现这些改进所产生的影响，通过提供 OCS 中全局行为的表示模型，其中包括因果关系和政策、安全方面，以及进步属性。

参考文献

1. T. Ahmed and A. R. Tripathi. Static verification of security requirements in role based cscw systems. SACMAT, pp. 196–203. ACM, 2003.
2. S. Bajaj, et al. Web services policy 1.2-framework (WS-policy). W3C Member Submission, 25, 2006.
3. C. Blundell, K. Fisler, S. Krishnamurthi, and P. Van Hentenryck. Parameterized interfaces for open system verification of product lines. In *ASE 2004, 19th IEEE International Conference on Automated Software Engineering, Linz, Austria*, pp. 258–267. IEEE Computer Society, 2004.
4. V. Braun, N. Kalt, B. Steffen, and T. Margaria. Hierarchical service definition. In Annual Review of Communication, pp. 847–856. International Engineering Consortium Chicago (USA), IEC, 1997.

5. J. Bredereke. On feature orientation and on requirements encapsulation using families of requirements. In M. D. Ryan, J.-J. Ch. Meyer, and H.-D. Ehrich, eds. *Objects, Agents, and Features, Volume 2975 of Lecture Notes in Computer Science*, pp. 26–44. Springer, 2003.
6. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 2001.
7. K. Fisler, S. Krishnamurthi, and D. J. Dougherty. Embracing policy engineering. In G.-C. Roman and K. J. Sullivan, eds. *FoSER*, pp. 109–110. ACM, 2010.
8. K. Fisler, S. Krishnamurthi, L. A. Meyerovich, and M. C. Tschantz. Verification and change-impact analysis of access-control policies. In G.-C. Roman, W. G. Griswold, and B. Nuseibeh, eds. *ICSE*, pp. 196–205. ACM, 2005.
9. C. Gordon, L. Meyerovich, J. Weinberger, and S. Krishnamurthi. Composition with Consistent Updates for Abstract State Machines. 2008.
10. P. Griffin. Introduction to XACML. Technical Report, Bea Systems, 2004.
11. H. Harris and M. Ryan. Theoretical foundations of updating systems. In *ASE*, pp. 291–294. IEEE Computer Society, 2003.
12. M. Hennessy and R. Milner. On observing nondeterminism and concurrency. In J. W. de Bakker and J. van Leeuwen, eds. *ICALP, volume 85 of Lecture Notes in Computer Science*, pp. 299–309. Springer, 1980.
13. F. Howar, B. Steffen, and M. Merten. From zulu to rers—Lessons learned in the zulu challenge. In *Proceedings of the ISoLA 2010 (1)—Leveraging Applications of Formal Methods, Verification, and Validation—Heraklion, Crete, Greece, Oct. 2010, Proceedings, Part I, volume 6415 of Lecture Notes in Computer Science*, pp. 687–704. Springer, 2010.
14. H. Hungar, T. Margaria, and B. Steffen. Test-based model generation for legacy systems. In *IEEE International Test Conference (ITC), Charlotte, NC, September 30–October*, pp. 971–980. IEEE Computer Society, 2003.
15. H. Hungar, O. Niese, and B. Steffen. Domain-specific optimization in automata learning. In *Proceedings of the CAV 2003, 15th International Conference on Computer Aided Verification, Boulder, CO, volume 2725 of Lecture Notes in Computer Science*, pp. 315–327. Springer, 2003.
16. H. Hungar and B. Steffen. Behavior-based model construction. *STTT, International Journal on Software Tools for Technology Transfer*, 6(1):4–14, 2004.
17. B. Jonsson, T. Margaria, G. Naeser, J. Nyström, and B. Steffen. Incremental requirement specification for evolving systems. In *FIW, Feature Interactions in Telecommunications and Software Systems VI, May 17–19, 2000, Glasgow, Scotland, UK*, pp. 145–162. IOS Press, 2000.
18. B. Jonsson, T. Margaria, G. Naeser, J. Nyström, and B. Steffen. Incremental requirement specification for evolving systems. *Nordic Journal of Computing*, 8(1):65–87, 2001.
19. S. Jörges, T. Margaria, and B. Steffen. Genesys: Service-oriented construction of property conform code generators. *ISSE*, 4(4):361–384, 2008.
20. M. Karusseit and T. Margaria. Feature-based modelling of a complex, online-reconfigurable decision support service. WWV '05, 1st International Workshop on Automated Specification and Verification of Web Sites, March 2005. ENTCS 1132.
21. M. Karusseit and T. Margaria. Feature-based modelling of a complex, online-reconfigurable decision support service. *Electronic Notes in Theoretical Computer Science*, 157(2):101–118, 2006.

22. M. Karusseit and T. Margaria. A web-based runtime-reconfigurable role management service. In *Automated Specification and Verification of Web Systems, 2006. WWV'06. 2nd International Workshop*, pp. 53–60. IEEE, 2007.
23. M. Karusseit, T. Margaria, and H. Willebrandt. Policy expression and checking in xacml, ws-policies, and the jabc. In *TAV-WEB 2008, Proceedings of the Workshop on Testing, Analysis, and Verification of Web Services and Applications, held in conjunction with the ISSTA 2008, Seattle, Washington, USA*, pp. 20–26. ACM, 2008.
24. S. Katz and G. Joseph. Aspects and superimpositions. In A. M. D. Moreira and S. Demeyer, eds. *ECOOP Workshops, volume 1743 of Lecture Notes in Computer Science*, pp. 308–309. Springer, 1999.
25. S. Krishnamurthi. The continue server (or, how I administered PADL 2002 and 2003). In V. Dahl and P. Wadler, eds. *PADL, volume 2562 of Lecture Notes in Computer Science*, pp. 2–16. Springer, 2003.
26. H. C. Li, S. Krishnamurthi, and K. Fisler. Verifying cross-cutting features as open systems. In SIGSOFT FSE, pp. 89–98, 2002.
27. D. R. Licata and S. Krishnamurthi. Verifying interactive web programs. In ASE, pp. 164–173, 2004.
28. T. Margaria. Service is in the eyes of the beholder. *Computer*, 40:33–37, 2007.
29. T. Margaria and M. Karusseit. Community usage of the online conference service: An experience report from three cs conferences. In *I3E 2002, 2nd IFIP Conference on E-Commerce, E-Business, E-Government (I3E 2002), towards the Knowledge Society: eCommerce, eBusiness, and eGovernment, Lisbon, Portugal, volume 233 of IFIP Conference Proceedings*, pp. 497–511. Kluwer, 2002.
30. T. Margaria, O. Niese, H. Raffelt, and B. Steffen. Efficient test-based model generation for legacy reactive systems. In *HLDVT '04: Proceedings of the High-Level Design Validation and Test Workshop, 2004. Ninth IEEE International*, pp. 95–100, IEEE Computer Society, Washington, DC, 2004.
31. T. Margaria, H. Raffelt, and B. Steffen. Analyzing second-order effects between optimizations for system-level test-based model generation. In *IEEE International Test Conference (ITC), Austin, TX (USA), November*, p. 467, 2005.
32. T. Margaria and B. Steffen. Lightweight coarse-grained coordination: A scalable system-level approach. *STTT*, 5(2–3):107–123, 2004.
33. T. Margaria and B. Steffen. Agile it: Thinking in user-centric models. In *Leveraging Applications of Formal Methods, Verification and Validation, Proceedings of the ISoLA 2008, volume 17 of Communications in Computer and Information Science*, pp. 490–502. Springer Verlag, 2009.
34. T. Margaria and B. Steffen. Continuous model-driven engineering. *IEEE Computer*, 42(10):106–109, 2009.
35. T. Margaria, B. Steffen, and M. Reitenspieß. Service-oriented design: The roots. In *ICSOC 2005: 3rd ACMSIGSOFT/SIGWEB International Conference on Service-Oriented Computing, LNCS N.3826*, pp. 450–464, Springer Verlag, Amsterdam, December 2005.
36. J. McCarthy and S. Krishnamurthi. Interaction-safe state for the web. Scheme and Functional Programming, 2006.
37. T. Moses, et al. Extensible access control markup language (XACML) version 2.0. Oasis Standard, 200502, 2005.
38. M. Müller-Olm, D. A. Schmidt, and B. Steffen. Model-checking: A tutorial introduc-

tion. SAS, pp. 330–354, 1999.

39. H. Raffelt, B. Steffen, and T. Berg. LearnLib: A library for automata learning and experimentation. In *Proceedings of the 10th International Workshop on Formal Methods for Industrial Critical Systems (FMICS '05)*, pp. 62–71, ACM Press, Lisbon, Portugal, 2005.
40. H. Raffelt, M. Merten, B. Steffen, and T. Margaria. Dynamic testing via automata learning. *STTT*, 11(4):307–324, 2009.
41. H. Raffelt, B. Steffen, T. Berg, and T. Margaria. LearnLib: A framework for extrapolating behavioral models. *STTT*, 11(5):393–407, 2009.
42. R. Sandhu, E. Coyne, H. Feinstein, and C. Youman. Role-based access control models. *Computer*, 29(2):38–47, February 1996.
43. M. Shahbaz and R. Groz. Inferring mealy machines. In FM, pp. 207–222, 2009.
44. M. Shaw and D. Garlan. *Software Architecture: Perspectives on an Emerging Discipline*. Prentice Hall, 1996.
45. B. Steffen and T. Margaria. Metaframe in practice: Design of intelligent network services. In *Correct System Design—Correct System Design, Recent Insight and Advances, volume 1710 of Lecture Notes in Computer Science*, pp. 390–415. Springer, 1999.
46. B. Steffen and T. Margaria. Business process modeling in the jABC: The one-thing approach. In J. Cardoso and W. van der Aalst, eds., *Handbook of Research on Business Process Modeling*, pp. 1–26. IGI Global, 2009.
47. B. Steffen, T. Margaria, A. Claßen, and V. Braun. Incremental formalization: A key to industrial success. *Software—Concepts and Tools*, 17(2):78, 1996.
48. B. Steffen, T. Margaria, R. Nagel, S. Jörges, and C. Kubczak. *Model-Driven Development with the jABC, Volume 4383 of LNCS*, pp. 92–108. Springer Berlin/Heidelberg, 2006.
49. R. Van De Stadt. Cyberchair: A web-based groupware application to facilitate the paper reviewing process, 2001. Available at: http://www.cyberchair.org

第 9 章 随机模型校验在工业中的应用：用户中心建模和 thinkteam 中的合作分析

Maurice H. Ter Beek
ISTI-CNR，比萨，意大利
Stefania Gnesi
ISTI-CNR，比萨，意大利
Diego Latella
ISTI-CNR，比萨，意大利
Mieke Massink
ISTI-CNR，比萨，意大利
Maurizio Sebastianis
Focus PLM 有限责任公司，费拉拉，意大利
Gianluca Trentanni
ISTI-CNR，比萨，意大利

9.1 简介

本章介绍了一种应用于系统设计阶段工业软件系统的建模和分析通用方法，也就是说，在系统实现之前，利用（随机）模型校验的方法。

在形式化建模和验证领域中的一个难点在于由于如相互交叉活动，会造成精细化模型往往产生巨大的状态空间问题。为此，本书所提方法具有某种不同性质。具体是开发和研究旨在解决具体问题的相当有限且抽象的形式化模型，并利用这些模型来验证可从易于用户理解的角度来观测系统模型各个方面相互依赖关系及其对系统性能影响的性能。这种采用模型校验的方式得到了类似于原型建模技术的支持。重点是以一种相对快捷的方式来获得一种形式化或某种程度上近似的对现有系统（设计）增加特定功能（定性和定量）的结果的理解。这与传统的将模型校验看作一种最大程度上保证复杂分布式算法正确性的已发展相当成熟的技术完全不同。从这个意义上来讲，本书所提出的模型校验方法有点类似于极限编程的思想[4]，即生成增加到系统（设计）中的简单 ad hoc 模型的新功能。经与工业伙伴 think3 的密切合作，采用本方法来建模和分析其产品数据管理（PDM）系统 *thinktem* 的扩展。从而，通过本方法隐式形成一种*动手实践*的经验

来阐明应用（随机）模型校验在工业中软件系统用户方面分析的发展潜力及其目前局限性。在工业中引入这种技术所面临的挑战是到目前为止模型校验尚不属于软件工程实践中的一部分。

产品生命周期管理（PLM）是指以一种最有效方式对公司产品从概念、设计、制造到销售和售后服务的整个生命周期的管理活动[5]。think3 的 PLM 是一种建立在 *thinkteam* 基础上的一套 PLM 综合应用程序，该 think3 的 PLM 应用程序正好迎合了制造工业中对产品设计过程的文档管理需求。*thinkteam* 使得企业能够以一种有效方式来收集、组织、自动处理和共享工程信息。研究中所进行的 thinkteam 设置中包含了所有与集中式关系数据库管理系统（RDBMS）交互的多个用户。RDBMS 系统控制着不允许执行文件编辑的称为库文件的类似文件系统知识库中的数据（类似于 CAD 文件）储存和检索。设计人员所采用的文件访问机制基于一种重复访问策略，即没有处理用户请求编辑文件权限的队列（或预订系统）。

在以前的工作中[6,7]，主要集中于利用定性模型校验来验证包含基于发布/订阅模式的用户通知系统的 thinkteam 性能。在此，分析了一些解决并发控制、准确性、可用性和用户认知的定性正确性。该研究使得在 thinkteam 中具有通知机制，然而一些对系统性能比对功能行为影响更大的可用性问题不能只利用定性模型校验进行分析。在文献［6，7］中提出过类似问题：表明由于其他竞争用户可更“幸运”地获得相同文档，则直接将该用户排除。事实上，这种行为可由用户只能采用一种基于重复的文件访问机制来进行解释。用户满意度取决于定性和定量两个方面。比如，频繁的重试可能会使用户很沮丧，所以应该避免发生这种情况。

定性模型校验分析会产生一个问题，但不会量化其对可用性的影响。在上述情况中，在获得文件之前用户所需进行的重试次数是一种重要的可用性测量。如果次数过多，就应考虑采用等待机制而不是简单地继续使用重试机制。通过随机模型校验对文献［8］中的设计选择进行了权衡分析，一种扩展的定性模型校验方法可允许对系统性能进行定性分析以及与性能和可靠性相关的分析（即定量分析）[3,9-12]。

一个基于模型的方法的问题是可扩展性，即如何处理通过对具有大量异步操作客户的交叉活动进行建模所产生的状态爆炸问题。在文献［13］中，研究了一种最近提出的基于模型的可扩展技术，即流体流动分析[14]。该技术可支持对具有通过同步形式协调的自发行为的多个重复实体进行分析。这是在进程代数方程基础上建立的，并对行为分析技术增加了定性分析。从形式化来讲，该技术包括一组通过利用性能评估进程代数（PEPA）[15]定义的规范来自动衍生的常微分方程（ODE）。通过标准数值方法求解的这组常微分方程的解有助于了解特殊状

态下组件聚合随时间的动态变化。该方法是从个体组件中抽象而得的。PEPA 测试平台可支持从 PEPA 规范中推导常微分方程组、常微分方程组的求解算法以及数值解的产生[16]。

最后，在文献［17］中，采用定量和定性的模型校验来评估 thinkteam，并提出 3 种扩展来用于该工业协作系统的设计。本章中，介绍了上述实验所用方法的通用方法。为详细阐述该方法，重新介绍了文献［17］中所提供的一种分析方法，即采用随机模型校验来验证具有多个重复库的扩展。这种验证是为了量化设计人员在存在多个重复库时下载/上传和重试所浪费的时间（并未记录）。为此，将采用与文献［17］中相同结构的随机模型，只是参数不同。更具体而言，就是采用对 think3 的一个客户实际使用 thinkteam 的日志文件进行分析而得到的实际参数。通过利用统计软件包 SPSS[18] 进行分析。由于参数是从真实数据中获得的，因此本章提出了一种对文献［17］中编辑和下载/上传时间更具通用假设的改进方法。由此，此处的贡献表明了模型校验对于探索性的设计阶段具有重要作用，不仅可以对不同设计方案进行对比，同时也对所提扩展方法的描述进行了完善和改进。

本章首先在 9.2 节中对 thinkteam 进行了详细介绍，并在 9.3 节中对日志文件进行了详细分析。在 9.4 节中，主要介绍了所提出的 thinkteam 扩展及其随机模型，然后对相关正确性和性能特性进行了形式化和验证，同时对输出结果进行了解释。9.5 节中包括了通过该工业案例分析所得到的经验教训。最后，在 9.6 中进行了小结。

9.2　thinkteam

本节对 think3 的 PDM 应用程序 thinkteam 进行概述，详细内容请参见网址：http：//www.think3.com。

设计过程中可产生和消化文档性信息（例如 CAD 图、模型和手册）和非文档性信息（材料账单、报告以及工作流记录）。这些信息的组合最终激活了生产实际对象的过程。信息管理不善往往能够直接影响产品生产阶段的成本结构。所以设计工作室的一个很重要的工作就是维护和更新之前发布的项目，即必须要从历史的角度来看过去的信息。这是 PDM 应用程序发挥作用的所在。

9.2.1　技术特点

thinkteam 是一个在 Wintel 平台上运行的 3 层数据管理系统。一种典型的安装形式是桌面客户端与一个集中式 RDBMS 服务器以及一个或多个文件服务器相互作用的网络。在这种设置情况下，每个客户端节点上的组件都可支持图形化界

面、元数据管理和集成服务。在RDBMS和文件服务器的基础上建模可实现持续性服务。接下来，简要概述与本章相关的thinkteam（逻辑）子系统的操作。

9.2.1.1　元数据管理

thinkteam允许用户管理具体实体（如文档和组件）的模型表示。利用由终端用户自定义的对象模型或元对象模型来描述这些表示（通常称为业务项目或业务对象），例如，通过改变属于不同类型对象的属性或增加对象类型。元数据管理是指在thinkteam中实现的对象实例的具体操作及其所遵循的规则。典型操作为创建、属性编辑（如增加/改变描述、价格）、修订、改变状态、与其他对象连接和删除。thinkteam利用RDBMS来保持和检索对象模型和在操作过程中创建的对象。RDBMS的交互本质上非常简单，且对终端用户完全可见。

9.2.1.2　仓库

在PDM应用程序中文档数据的受控存储区和检索一般称为仓库，库是一个类似文件系统的资源库。库的两大主要功能是①提供一个PDM应用程序控制的文档所管理的独立、安全的受控存储环境；②防止文档库的不一致更新或改变，而仍然允许符合业务规则的最大程度访问。第1个功能受到库系统底层实现的影响，而第2个功能是在用户可用的一组thinkteam操作（见表9.1中所列）协议下实现的。

表9.1　thinkteam用户操作

操　作	结　果
get	从库中抽取一份文档的只读备份
import	将一份外部文档插入库
checkOut	从库中抽取一份文档的备份（排他性）
unCheckOut	取消上次checkOut的结果
checkIn	替换库中的一份编辑（之前可checked out）文档
checkInOut	替换库中的一份编辑文档（重新获得编辑权限）

值得注意的是，文档访问（通过checkOut）基于一种重试策略，即没有处理文档编辑权限请求的队列系统和预订系统。

9.2.2　thinkteam的工作过程

thinkteam可在一个给定项目的整个产业化部分的各个设计阶段支持CAD设计人员。接下来简要介绍在建模阶段CAD设计人员最常用的库功能。

9.2.2.1　几何信息检索

在制造业中（thinkteam的首要目标）最常用的设计工作涉及更复杂产品的组件生产。描述这些产品的CAD图称为*装配*，且结构化为由多个（甚至上千

个）个体模型文件组成的组合文档。大多数几何数据是设计人员通过参考材料得到的，即实际创建或修改的组件周围的部分。设计人员需要利用这些参考材料来与正在进行装配的部分定位、适应和匹配。大多数参考组件都是受 PDM 控制的生产项，其复件（模型文件）存放在库中。设计人员访问这些参考组件的逻辑操作 get 操作，该操作可在浏览文件时自动执行。这是最常用的活动类型，且与下列所有其他活动以及上述未明确提出的许多其他活动（如可视化、打印）都相关。

9.2.2.2 几何模型修正

对现有组件进行修正是设计人员最常用的第 2 种操作。由于该组件是已经存在且受管理的（即存在于库中），因此设计人员以 checkOut 操作来显式表示对其修改以防止其他用户试图修改（独占锁）。在发布其设计之前，设计人员必须通过显式的 checkIn 操作将其发布到系统中，由此可使得其他设计人员修改该模型。若设计人员在修改模型时改变其想法，则应采用 unCheckOut 操作以使得解锁该模型并不保存 checkOut 操作之后的任何改变。最后，设计人员采用 checkI-nOut 操作来将模型的中间版本发布到库，并保持专有的修改权限。

9.2.2.3 几何模型创建

最后设计人员需创建一个完整的新组件模型，并将其加入系统中。由于该模型最终是在系统库之外创建的，因此需要执行 import 操作来将其在 thinkteam 中注册。

9.3 thinkteam 日志文件分析

thinkteam 可对 20 ~ 100 个用户处理数十万个日志文件。为获得 thinkteam 用户在特定应用情况下产生的实际数据，think3 可提供一个由某个制造业 2002 ~ 2006 年期间使用 thinkteam 的所有活动（9.2.1.2 节所列操作）组成的纯净的日志文件。为精细整定下节中介绍的 thinkteam 的特定模型参数，在此应重点关注以下以使用为核心的信息：编辑平均持续时间、文件未用于编辑的平均持续时间以及用户未成功获得编辑文件的平均次数。本节对此进行分析的目的是要了解编辑持续时间和文件占用的时间问题。

尽管该日志文件已被清理（即去除与上述操作无关的登录信息），但为达到特定目的，应对其进行进一步“清理”。由于数据量庞大，因此应关注于所采用的特定形式化以及以使用为中心的具体定量信息，结果表明开发专用脚本是执行该“清理”的最有效方式（大多数多重关系数据挖掘方法都需要加载整个主存储器中的数据，因此这不适用于挖掘包含大量数据的一个日志文件）。开发这些脚本还有助于在发现日志中的一些违规行为并自动纠正。这些日志中的错误主要

是包括由具有秒级精度的日志粒度所造成的错误操作顺序。采用更高的精度（如接近毫秒级）可避免纠正这些错误顺序。在此，采用相同的脚本开发技术来简化日志文件（如滤除无用的用户行为等），并将其转化为统计分析软件包 SPSS[18] 所接受的格式。

对于每个操作，所生成的日志文件中都包含其产生时间（以日-月-年以及小时-分钟-秒的格式）、执行操作的用户名以及操作所对应的库中文件。通过这种方式，日志文件中的每一行表示对库的一个原子级访问。虽然日志文件的格式易于处理，但该日志文件中包含有极为庞大的数据量（104 个用户对于 183492 个文件访问了 792618 次）。此外，think3 已经对不同年份之间的日志机制进行了改进，这就意味着难以比较不同年份所产生的日志文件，且 2006 年的日志是最完整的。正是由于这些原因，因此在分析时对所使用的日志文件限于 2006 年。

2006 年的日志数据包括了 83 个用户合作产生的总共 181535 个文件，其中 23134 个文件至少在 2006 年内检查过一次。其他文件专门作为参考材料使用，如通过 get 操作以只读方式下载。总共有 65 个用户参与了编辑。在此给出与下节中所要分析模型直接关联的数据子集的分析。这些分析包括编辑持续时间（即同一用户对某一文件执行 checkOut 和 checkIn 操作之间所需的时间）和文件未锁存的持续时间（即不同用户对同一文件执行 checkIn 和 checkOut 操作之间所需的时间）。而用户对某一锁存文件未成功执行 checkOut 的次数（即由其他用户签出）并未显式记录，由此不能直接判断。然而，可通过一种非直接的对研究期间修改同一文件的用户数近似来获得。为表明该思想，图 9. 1 中给出了一周内对某一特定文件的编辑活动。由图可知，4 个用户在一周内对同一个文件进行了 20 次编辑。

图 9. 2 和图 9. 3 给出了从日志文件中去除无关操作后的数据，这些操作类似于大量的 get 操作（这是最常用的操作），是通过系统管理员介入所产生的操作以及一些异常的日志操作。经过平均削减 40% 的计算后通过 SPSS v15. 0[18] 绘制出这些图（即去除最低值和最高值 20% 的值后对剩余值取平均）。在统计学中，削减后的均值比数据中存在野值的均值（即极值）更具有鲁棒估计意义，因为这对于野值的敏感性较低，同时仍能给出一个集中趋势或均值的合理估计。图 9. 2给出了 $N=37524$ 次编辑期间的分布直方图。x 轴为时间（单位为秒），从而每一条柱状包含位于相应间隔约 5min（278s）的数据集中的值。该直方图显示出除去最低值 20% 和最高值 20% 的所有编辑时间，实际上意味着去除了小于 111s（即约两 min）和大于 22256s（即约 370min）的编辑。由此可看到，一次编辑削减 40% 后的平均持续时间为 2657s，约 44min。由此可知，大多数编辑往往都很短。

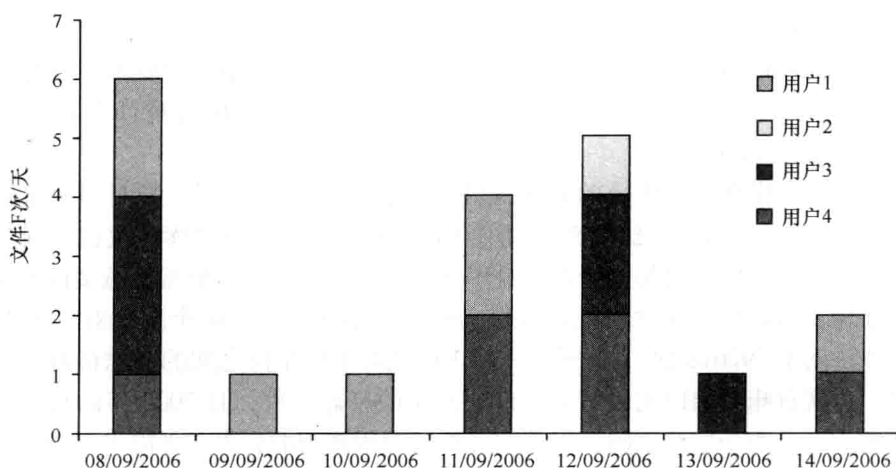


图 9.1 一周内某一文件的 checkOut 次数

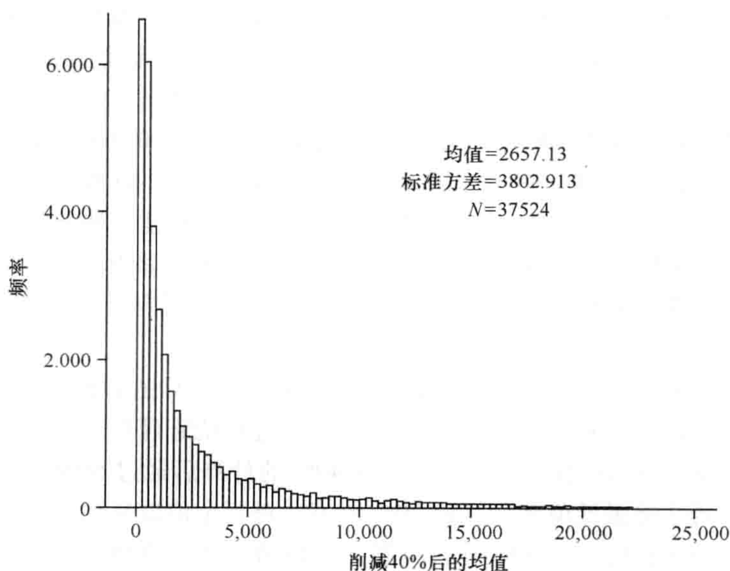


图 9.2 编辑持续时间

在图 9.3 中，显示了在至少两次编辑未锁存（即在 checkOut 操作中可获得）所涉及的 $N=23634$ 个所有文件时间间隔的直方图。这些数据给出由不同用户访问库所花费的时间。x 轴表示时间（单位为秒），每条柱状包含了位于大于 17h（625000s）的相应时间间隔内数据集中的值。大部分持续时间都位于最初的几个区间，这表明许多情况下这些文件的使用是相当集中的。

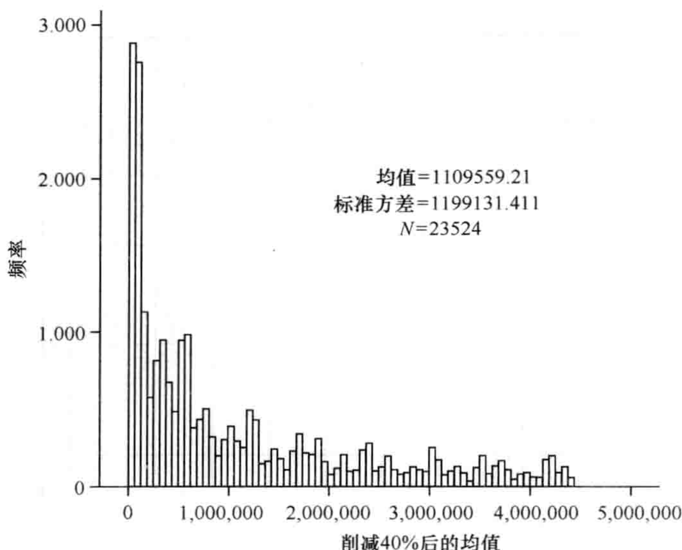


图 9.3　文件内部访问时间

从清理日志文件中提取了一个编辑持续时间的有序列表。由于具有 21365 个不同的持续时间，在图 9.4 中仅显示了部分有序列表（完整表格应包含 21365 行）。然而，该提取列表可清晰地表明在这组编辑持续时间列表中存在野值。在其中的一个极值，在不到 1s 的时间内修改 3962 个文件，而在另一个极值，在大约 300 天的时间内修改了几个文件。实际中，极短时间的编辑并不是对应于真正的用户编辑，而是对应于所记录的系统自动操作，且在日志文件中的特点是 checkOut- checkIn 时间很短。另一方面，极长时间的编辑是由于系统管理员为锁存文件所导致的（如用户忘记对文件进行 checkIn 操作）。显然，这两种极值情况都不符合典型的用户操作，这正是需要进行修剪的原因。

最后，表 9.2 中显示了在 2006 年由多个用户编辑的文件个数。对这些文件进一步分析表明在同一天多个用户对同一文件进行编辑非常普遍（见图 9.1），甚至在同一天有 8 个用户访问同一文件。值得注意的是，日志文件的分析只涵盖某个 think3 的特定客户所收集的 1 年数据，因此，这并不能完全代表 thinkteam 的一般使用情况。然而，记录的数据是可实际观察的，由此可提供一个 think-team 示例使用的信息，这有助于判断建模结果是否正确。另一方面，该日志文件的分析还表明有助于可用性问题进一步评估的数据目前尚未收集在 thinkteam 的日志文件中。因此，需继续合作以开发今后在日志文件中记录更多数据的进一步工作。

#	持续时间/s	持续时间 /(dd/hh/mm/ss)	频率	百分比	累积百分比
1	0	00d 00h 00′ 00″	3926	6.2751%	6.2751%
2	1	00d 00h 00′ 01″	1553	2.4822%	8.7574%
3	2	00d 00h 00′ 02″	248	0.3963%	9.1538%
4	3	00d 00h 00′ 03″	98	0.1566%	9.3104%
5	4	00d 00h 00′ 04″	64	0.1022%	9.4127%
⋮	⋮	⋮	⋮	⋮	⋮
1064	1063	00d 00h 17′ 43″	8	0.0127%	49.960%
1065	1064	00d 00h 17′ 44″	4	0.0063%	49.966%
1066	1065	00d 00h 17′ 45″	9	0.0143%	49.980%
1067	1066	00d 00h 17′ 46″	4	0.0063%	49.987%
1068	1067	00d 00h 17′ 47″	9	0.0143%	50.001%
1069	1068	00d 00h 17′ 48″	12	0.0191%	50.020%
1070	1069	00d 00h 17′ 49″	4	0.0063%	50.027%
1071	1070	00d 00h 17′ 50″	8	0.0127%	50.039%
1072	1071	00d 00h 17′ 51″	12	0.0191%	50.059%
⋮	⋮	⋮	⋮	⋮	⋮
21361	23733879	274d 16h 44′ 39″	1	0.0015%	99.993%
21362	24787896	286d 21h 31′ 36″	1	0.0015%	99.995%
21363	25150004	291d 02h 06′ 44″	1	0.0015%	99.996%
21364	25484279	294d 22h 57′ 59″	1	0.0015%	99.998%
21365	25900836	299d 18h 40′ 36″	1	0.0015%	100.00%

图9.4 编辑持续时间的有序列表

表9.2 至少两个以上用户编辑的文件个数

文件数	5077	1407	301	79	24	22	8	8	6	10	3	1	1	1
用户数	2	3	4	5	6	7	8	9	10	11	12	13	14	17

9.4 具有复制仓库的thinkteam

接下来，在本章随后的章节中将介绍think3所采用的最新thinkteam扩展的研究成果，即增加多次复制库。这些库位于一些地理上分布的位置（见图9.5，其中checkIn/Out文档由虚线箭头表示，而元数据操作由实线箭头表示）。然而，thinkteam很清楚复制库的状态以及所有文件的状态，即当前是否有设计人员检查或用于修改的文件。

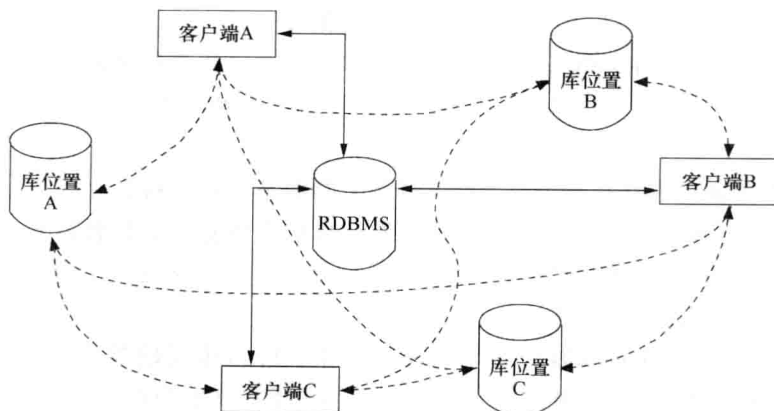


图9.5 具有复制库的thinkteam

当设计人员在新的配置中查询thinkteam时，如复制文件，thinkteam通常会通过赋值“可能最佳”的库位置来响应。理想情况下，这是设计人员首选的库位置（在带宽上具有最佳连接），但如果首选位置被占用或工作量太大时，会赋值一个次优位置。另一方面，如果thinkteam注意到所需要文件的最近checkin均是由同一设计人员执行，则thinkteam将会提示设计人员可以在桌面上使用该文件的本地版本（从而可省去checkOut操作）。

如果设计人员已获得一个库的地址，则该设计人员可checkOut该文件，并进行编辑，最终checkIn该文件，然后通过其首选的库位置进行再一次循环。每次checkIn之后，相应的位置会提示thinkteam文件已上传。之后，thinkteam更新文件状态，即解除文件锁存，使之可被其他设计人员使用。这种通信方式还可对thinkteam传输库位置的状态信息。无论是需保持一致的库位置之间，还是图

9.5 中的库位置和 thinkteam 之间都不需要通信。在本章中所考虑的 thinkteam 模型中，并未明确解决库之间的通信问题，而是假设通过合适算法保持一致。库和 thinkteam 之间的通信方式需要进行明确建模。

9.4.1 thinkteam 的随机模型

假设一个由 3 个库位置（Va、Vb 和 Vc）构成的模型中包括相同的文件资源库、3 个竞争同一文件的明确建模的客户端（CA、CB 和 CC）以及 thinkteam 应用程序 TT。每个库位置与 TT 应用程序相连，且定期向 TT 发送其状态。出于性能分析的目的，应关注库位置的状态，如工作量、可用性（即上传或下载）以及为不同客户端所提供的带宽。TT 保持对所有文件的状态进行记录，其中包括如文件是否锁存（即由通过客户端检查）或是否可用于下载和修改。

9.4.1.1 假设条件

该模型取自文献［17］，且基于以下假设条件：

1）客户和库之间的带宽恒定，且每个客户首选最佳连接来从库中下载和上传文件。然而，有时首选连接可能连接不上，在这种情况下，客户将使用次优库。

2）每个客户都具有一个表示库选择顺序的静态优先权列表。

3）3 个明确建模的客户不会对整个系统的整体性能产生显著影响（包括许多由不同库的响应特征隐式建模的活跃客户端）。目的是从这 3 个客户的角度来分析系统的可用性和正确性。

4）只考虑 thinkteam 客户可用的操作子集，即最重要的操作：checkOut 和 checkIn。这样会使得模型相对简单，同时可易于增加其他操作。

5）目前尚不允许 TT 来提示客户可利用桌面上该文件的本地版本，如果 TT 注意到所需文件的最近 checkIn 操作是由同一客户执行。

这些假设来源于 think3 中有关合理客户行为（1），thinkteam 抽象模型建模需要（2、4 和 5）以及严格的逻辑推理（3）的讨论。

9.4.1.2 模型

模型是由 PEPA[15] 指定的，在此不再详细解释。在文献［17］的附录中给出了完整的 PEPA 规范。为简洁表述，在此将客户、库和 TT 描述为图 9.6 ~ 图 9.8 中的一种随机自动机，这也曾用于与 think3 的同事进行的讨论中。在讨论模型分析时，状态和迁移的标签在下节中具有重要作用。迁移标签的一般形式是“from_to_action”，其中“from_to”部分表明进程间的信息流方向（如 CA_TT 表示从 CA 到 TT 的通信），而“action”部分表明一种特定操作（如 cO_s 表示 checkOut 成功，cO_f 表示 checkOut 失败，cI 表示 checkIn）。

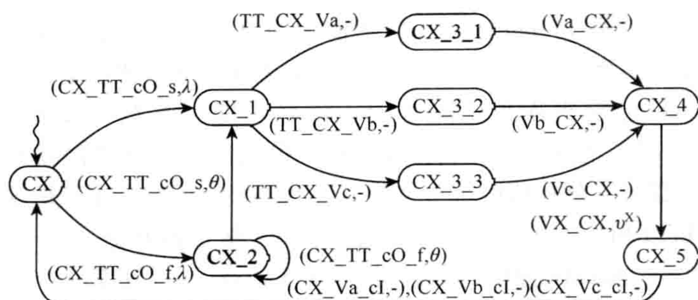


图 9.6　X = A，B，C 时的客户端 CX

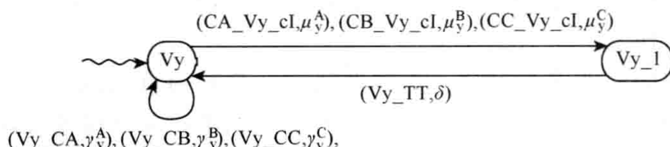


图 9.7　y = a，b，c 时的库 Vy

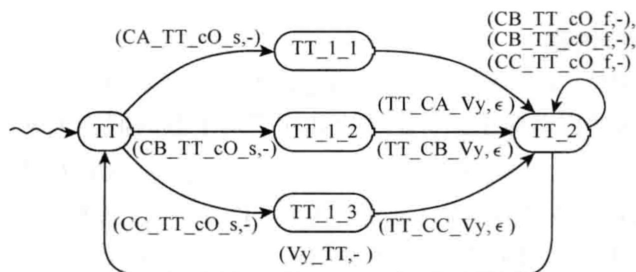


图 9.8　y = a，b，c 时的 TT

9.4.1.3　客户端进程

客户端 CX（其中 X = A，B，C）的行为建模如下（见图 9.6）。最初在 CX 状态下以速率[⊖] λ，客户端 CX 向 TT 发送一个请求来下载需修改的文件。在文件可用情况下，向 TT 发送请求成功（CX_TT_cO_s，λ），而当文件正由其他客户端编辑时，则向 TT 发生请求失败（CX_TT_cO_f，λ）。如果请求成功，TT 向 CX 提供“可能最佳”的库位置地址（如 TT_CX_Va 表明 CX 接收到库 A 的地址）。

在当前模型中为库地址赋值的策略非常简单，即每个客户以最大静态概率接收其首选库位置地址（即优先级列表中的第一个地址），而以较小概率接收其他不同的库位置地址。事实上，首选库位置并不是总能获得，这是由于工作量过大或暂时不可用。为了更好地与系统性能特征相匹配，可以对这些概率进行调整。

⊖ 一个行为的比率 r 定义了平均持续时间给定为 $1/r$ 时的持续时间 Δt，这是由于根据定义，Δt 是与比率 r 相关的指数分布的随机变量。

理论上，可通过分析在 thinkteam 中当前使用的单个库的性能特征的日志文件来获得概率表示。然而，在已分析的这些特定日志文件中并未注册该信息。

当客户 CX 已获得库位置地址时，该客户即可下载（如从库 A 中通过（Va_CX，-））所需文件（一般是由要编辑文件构成的部分文档或不进行编辑的组件周围相关部分文件构成的部分文档）然后在状态 CX_4 下*编辑*文件，并为特定客户的编辑以相应平均时间 $1/v^X$ 的比例 v^X 离开该状态（通过（CX_CX，v^X）），最后通过 checkIn 操作（如通过（CX_Va_cI，-）到库 A）将文件上传到库，接下来根据优先权列表下载，并返回到初始状态 CX。带“-”比率的行为是被动的，即在同步过程中（本例中为客户和库进程之间的同步）建立比率值，其中后者确定该值。

如果客户的文件请求失败，则首先执行一系列的重试操作（在状态 CX_2 下），然后获得文件。这本质上意味着客户以较大的比率 θ 继续发送请求（通过（CX_TT_cO_f，θ）。在经过一系列的连续请求失败后，客户最终请求成功，并移动到状态 CX_1。

9.4.1.4 库运行进程

库 Vy（其中 y =a，b，c）的行为建模如下（见图 9.7）。库（位置）可以接受由客户发送的以对应于该特定客户和库平均下载时间为 $1/\gamma_y^X$ 的速率 γ_y^X 的下载请求（通过（Vy_CX，γ_y^X）），且以速率 μ_y^X 的 checkIn 操作来改变。每次 checkIn 操作之后，库提示 TT（通过（Vy_TT，δ））文件已*上传*。通过这种方式，TT 可及时更新文件状态，即解除锁存，使之可以供其他客户使用。同理，以相同的通信方式也可对库状态信息传递给 TT 进行建模。

9.4.1.5 TT 运行进程

TT 的行为建模如下（见图 9.8）。最初，TT 等待处理客户发送的文件请求（如通过行为（CA_TT_cO_s，-）来自客户 A）。在文件请求成功的情况下，TT 通过上述赋值策略来将库赋值给客户。

该策略可通过不同赋值之间的竞争关系随机建模如下。如果客户 A 赋值于库 A 的平均概率约为 50%，赋值于库 B 的平均概率约为 33%，赋值于库 C 的平均概率约为 16%，则可通过选择合适的概率来表征。例如，状态 TT_1_1 具有 3 种输出迁移，分别标记为（TT_CA_Va，300）、（TT_CA_Vb，200）和（TT_CA_Vc，100）。因此，从状态 TT_1_1 总的退出比率为 300 + 200 + 100 = 600，则客户 A 赋值于库 A 的概率为 300/600 = 0.5。这样相对较大的退出比率表明相对于其他操作库的赋值操作是非常快的。因此，模型参数的设置不会显著影响模型的性能分析（由于空间和便于阅读的原因，在图 9.8 中部分显示这些细节，即仅给定了相对速率的表示方法 ε）。行为（TT_CX_Vy，ε）表示赋值一个库，并锁存文件，同时发送给客户一个库地址。

表9.3 比率值

λ	$\gamma_a^A=\mu_a^A$	$\gamma_a^B=\mu_a^B$	$\gamma_a^C=\mu_a^C$	$\gamma_b^A=\mu_b^A$	$\gamma_b^B=\mu_b^B$	$\gamma_b^C=\mu_b^C$
0.1	8	4	6	6	8	4

任何对同一文件的继续请求都会明确拒绝（如客户A通过（CA_TT_cO_f，-）），直到TT接收到表明文件已上传（如库Y通过（Vy_TT，-））的来自库的消息。然后TT恢复到其初始状态，准备接受对文件的下一个请求。

9.4.1.6 完全规范

thinkteam的规范是利用3个客户进程、3个库运行进程和TT运行进程的PEPA合作运算符（在此用‖表示）通过并行组合来实现的，具体描述如下：

用户 $\|_{CX_TT_cO_z, TT_CX_Vy, CX_Vy_cI, Vy_CX}$ 系统

式中，用户 = CA‖CB‖CC；X = A，B，C；y = a，b，c；z = f，s；系统 = TT $\|_{Vy_TT}$（Va‖Vb‖Vc）。

值得注意的是，上述分析仅限于3个客户在几乎同一时间内对同一文件竞争的一个模型。正如之前解决具有一个中心库的thinkteam模型一样[8]，该模型也可很容易地扩展到具有有限个明确建模的客户。采用基于仿真的模型校验方法[19]可分析大量客户的情况，但会产生较高的计算成本且结果精度降低。

9.4.2 随机模型分析

本章采用了概率符号模型校验器PRISM[11,20]，可支持包括在连续时间马尔科夫链（CTMC）上的连续随机逻辑性能验证（CSL）[21]。CTMC可通过高级描述语言生成，其中包括PRSIM为其提供前端的PEPA。具有3个客户的完整PEPA规范（在文献［17］的附录中给出）可产生具有104个状态和330个迁移的CTMC。本节所介绍的所有分析都是在PRSIM v3.1.1中执行的，且占用极少的CPU时间。用于概率计算的数值迭代方法是精度为10^{-6}的Gauss-Seidel法。详情可参见http://www.prismmodelchecker.org或文献［20］。

$\gamma_c^A=\mu_c^A$	$\gamma_c^B=\mu_c^B$	$\gamma_c^C=\mu_c^C$	$V^A=V^B=V^C$	θ	$\delta=\varepsilon$	$\varepsilon_2=2\varepsilon$	$\varepsilon_3=3\varepsilon$
4	6	8	1.35	6	100	200	300

利用与文献［17］具有相同结构的随机模型来验证重要参数（即编辑和下载率）不同假设条件下的各种性能。通过实际应用thinkteam可经验性地获得这些假设，即利用在9.3节中具体工业实际应用thinkteam所产生的日志文件分析结果以及与think3的详细讨论后得到。值得注意的是，上述分析只是针对一年内只有一个用户使用thinkteam。剩余的比率值（如重试率）由think3对进行估计，但并未包括在本实验的日志文件中。在表9.3中列出所有比率值。下载率可以从

库（客户）名称的下角（上角）字母中获得。例如，γ_b^A表示库 B 和客户端 A 之间的下载率。

在此选择模型的时间单位为 1h。例如，比率 $\gamma_b^A = 6$ 表明库 B 和客户端 A 之间的平均下载时间为 60/6 = 10min。该值看起来可能较大，但正如在 9.2.2 节中所解释的，CAD 设计人员是使用 thinkteam 来共享装配组件，即组合文档是由多个个体模型文件（甚至几千个）组成的。因此，当客户需要修改某个文件时，必须下载（形成该文件的上下文）大量其他文件。例如，比率 $\nu^A = 1.35$ 表明客户 A 在编辑文件时需花费 60/1.35 ≈ 44min，这实际上对应于 9.3 节中提到的编辑平均持续时间。

9.4.2.1 定性性能分析

在分析模型性能之前，获得其功能正确性的信任度是很重要的。在此，需要验证定性性能，如消除死锁、可进展属性和文件编辑权限的互斥。针对具有 PRSIM 的 PEPA 模型进行随机模型校验时，首先要利用 PRSIM 的前端设备将 PEPA 模型转化成等效的 PRSIM 模型，然后对后者进行 CSL 公式验证。这些公式中的变量为 PRISM 模型中的状态。

最终死锁时的概率应（大多数情况下）为零。该属性可通过预定义的 PRISM 标签“死锁”在 CSL 中形式化，其中，对每个吸收状态（如退出概率为 0 的状态）通过以下方式进行标记：

$$P_{\leqslant 0}([\text{ true } U \text{ “deadlock” }])$$

此时，PRISM 验证适用于本模型。

只要客户 X 成功地从库 A 中 checkOut 一个文件，则最终要对该文件执行 checkIn 操作。该属性可通过下列 CSL 公式来得到：

$$P_{\leqslant 0}([\text{ true } U \text{ “CXcheckIn” } \{\text{“CXcheckOut”}\}])$$

其中，标签“CXcheckIn”定义为 CX_STATE = CX_5，而“CXcheck-Out”定义为 CX_STATE = CX_3_1。PRISM 验证上述公式适用于本模型。

两个客户最终同时获得对同一文件修改权限的概率应（大多数情况下）为零，该属性可通过下列公式形成：

$$P_{\leqslant 0}([\text{ true} U(\text{“OkAB” OR “OkAC” OR “OkBC”})])$$

其中，XY = AB，AC，BC 时，标签

“OkXY” = (CX_STATE = CX_1) AND (CY_STATE = CY_1)

意味着两个客户端（X 和 Y）同时获得编辑文件的权限（即分别位于 CX_1 和 CY_1 状态）。PRISM 验证该公式适用于本模型。

9.4.2.2 定量性能分析

本节将介绍本模型的性能问题，特别是从客户端角度来看的一些可用性问题。

快速释放文件

上节中的第2个公式只表明客户端最终可上传文件（下载文件之后）。通过下列 CSL 公式可将该问题量化：

$$P_{=?}([\text{true } U^{\leqslant 1}\text{“CXcheckIn”}\{\text{“CXcheckOut”}\}])$$

即，在下载（通过 checkOut）文件（状态 CX_3_1）后的1h 内，客户端 X 在状态 CX_5 准备上传（通过 checkIn）文件的概率是多少？在表明编辑率 ν^X 从0.75变化到5，即编辑文件的平均时间为 12 ~ 80min（对应于图 9.2 中分布平均值的半标准方差内的值）的图 9.9 中给出上述结果。正像预期的那样，客户编辑的时间越短，则在1h 内获得结果的概率越大。

长期运行行为

影响客户端使用 thinkteam 以及不同活动所需时间的一个参数是访问同一文件时所用的平均时间。图 9.10 中给出了改变内部访问时间时，某一特定客户端在不同活动上花费的平均时间的变化，即对于每个活动，可观测到客户端 X 在显著影响请求率的不同 λ 值下所花费时间的百分比。

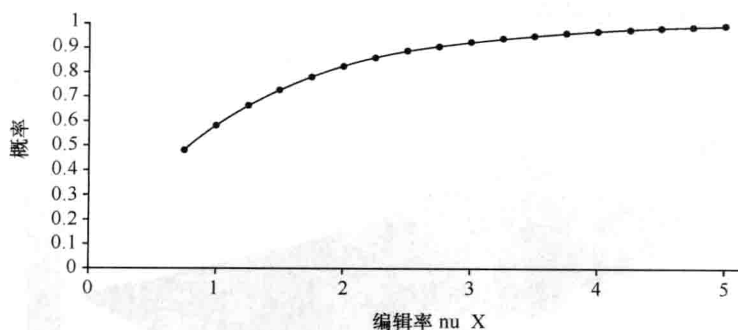


图 9.9　客户端在执行 checkOut 后 1h 内 checkIn 该文件的概率

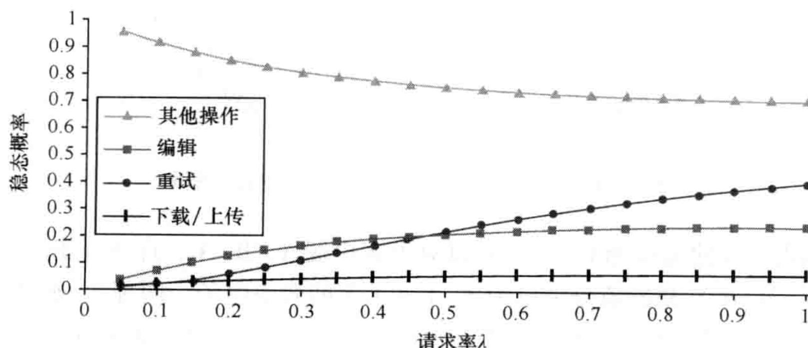


图 9.10　对各种文件内部访问进行不同活动时客户端所花费的时间

由图可知，在 λ 很小的情况下，大部分时间都花费在除编辑、重试和下载/上传之外的其他操作上。随着 λ 的增大，这种模式会发生很大变化。等待文件（重试）所花费的时间迅速增大，从而客户端花费在其他操作上的时间更少。同时，客户端还需在编辑上花费更多时间，但该时间增大的并不快。值得注意的是，从某一特定点开始（当 $\lambda=0.45$ 时），客户端用于重试获得文件的时间要大于实际编辑文件的时间。

通过分析获得图 9.10 中结果的属性可在 CSL 中简化为如下所示的稳态特性形式化：

$$S_{=?}([\text{“Client XinStateZ”}])$$

其中，标签“Client XinStateZ”由表示所关心的特定活动的状态代替，例如，CX_STATE = CX_2 表明客户端 X 正重新尝试获得文件。

下载和上传文件所需时间

客户端下载和上传文件所需时间很大程度上取决于与库连接的带宽、文件大小以及库的工作量。图 9.11 中给出了库 A 和 B 的工作量改变对客户端 A 下载和上传文件所需百分比时间的影响。

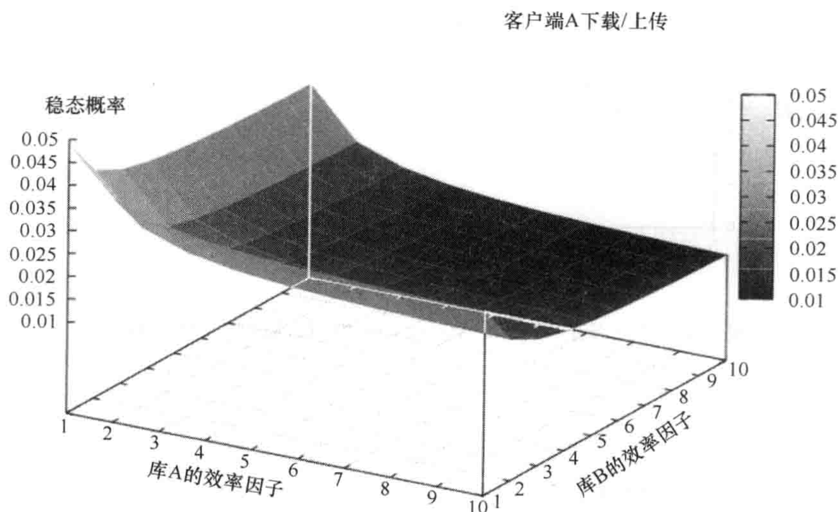


图 9.11　客户端在下载或上传文件时的稳态概率

x 轴表示变化范围为 1 ~ 10，乘以对于客户端 A、B、C，库 A 下载率 γ_a^X（其中 X = A，B，C）初始值分别设为 2、1 和 1.5 的效率因子。同理，y 轴表示乘以对于客户端 A、B、C，库 B 下载率初始值分别为 1.5、2 和 1 的效率因子。所有其他比率值见表 9.3。

因此，效率因子越高，库的执行速度越快。实际上，正如预期的那样，可以看到当库（A 和 B）最优工作时，客户端 A 长期上传和下载时所花费时间的概率最小。同时还观察到，如果库 B 的工作量较大，则执行速度较慢，从而将会影响库 A 上传和下载所需的时间。这是因为客户端 A 从库 B 下载（上传到）库文件 B 所需的部分时间。另一个观察是在库已达到某种性能时，该百分比时间的减少程度不大：对于分析过程所选择的参数设置，这通常发生在效率因子为 5 时。通过验证客户端 A 位于状态 CA_3_1、CA_3_2、CA_3_3 或 CA_5 时的上述公式可得到如图 9.11 所示的结果。

请求成功的重试次数

thinkteam 的可用性还取决于客户端能多久无法获得所要修改的文件。无法获得所需文件意味着客户需要在随后的阶段中需要继续花费时间来获得文件或更改其工作计划。如果这种情况频繁发生，则可能导致客户厌烦，同时也会浪费时间。另外，还可能导致错误产生（用户可能后来忘记编辑文件或忘记所进行的修改）或在整个工作流计划中产生问题，以及推迟最终产品的交付。因此能够对不同条件下以及 thinkteam 不同用户配置下产生的问题进行量化会有很大帮助。例如，不同的设计阶段可能会产生 thinkteam 的不同使用：最初，客户端可能需要较多的时间来修改文件，因为这是一个全新的设计，但当接近最后期限时，可能会是许多客户端需频繁进行小修改以实现不同组件微调的阶段（见图 9.1）。

图 9.12 中给出了一个分析结果。其表明客户端 A 获得修改文件权限所需的重试平均次数会随着所有客户端的编辑和重试率同时增大（导致平均编辑时间更短且重试更加频繁）以及客户端需要文件的频率增大（即表明上述客户行为接近最后期限）而变化。

编辑和重试率的初始值分别设为 0.5 和 1，其中乘以一个变化范围为 1 ~ 10 的因子，而请求率（其倒数为 checkIn 某一文件以及随后修改同一文件之间所花费的平均时间）的变化范围为 0.05 ~ 0.3。因此，一方面认为，平均客户端行为从 2h 的编辑文件并每隔 1h（x 轴的左端）重新尝试 checkOut 文件，到编辑文件仅需 12min 并每隔 6min（x 轴的右端）重新尝试 checkOut 文件。另一方面，checkIn 文件以及随后请求 checkOut 同一文件所花费的时间，从每天大约一次（y 轴的左端）到每 3h 一次（y 轴的右端）。

通过扩展本模型的 PRISM 规范，利用一个回报结构来计算成功请求次数以及另一个回报结构来计算重试次数来得到图 9.12 中的结果。回报结构是在需计算行为的 CTMC 定义基础上来定义的。从而，可通过一些基于回报的特性来分析模型。在此，采用计算长期运行平均回报的稳态回报公式。该公式在 PRISM 中可形式化为

$$R\{\text{“label”}\} = ?\ [S]$$

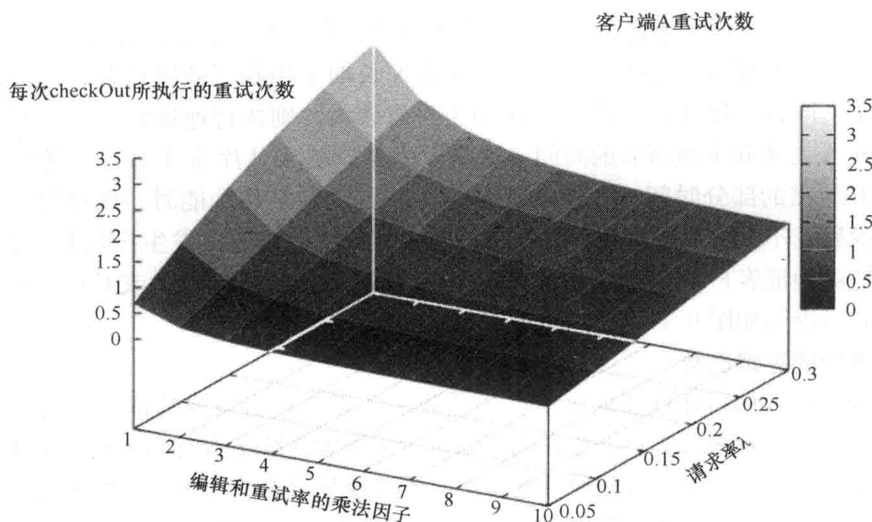


图 9.12　客户端执行重试获取某一文件的平均次数

式中，“label” 为回报结构；S 表示所计算的稳态回报。每次请求成功所重复的请求失败次数如图 9.12 所示，这是通过将 label = NrFailedRequestsClientA 时的公式输出结果除以 label = NrSuccessfulRequestsClientA 时的结果而得到的。

由图 9.12 可知，当客户端编辑文件的平均时间越长而试图重新尝试 checkOut 文件并不频繁（朝 x 轴的左端），特别是 checkIn 文件和随后 checkOut 文件的请求同时减少（朝 y 轴的右端）时，客户端需要执行获取文件的重试次数将会增大。

在一个特定应用环境下具有更详细的数据可提高采用相同模型进行随机分析的准确度。在实验环境下 thinkteam 的具体应用特点表明每次成功 checkOut 时平均重试次数是可接受的，同时还表明当编辑和重试率在许多客户端需要频繁对文件进行微小修改的接近最后期限期间内逐步增大时，平均重试次数也是可接受的。

在假设编辑时间更长（大约 200min 而非 44min）且下载/上传更快（大约 1min 而非 10min）的不利情况下，对与文献［17］中相同的模型进行分析表明，每次成功时重试次数可能会高达无法接受（大约是图 9.12 中重试次数的 10 倍）。在这种情况下，等待队列策略可能会比重试策略更加适合，从而可减少客户端重试获得文件所浪费的时间。通过采用不同分析方法对文献［13］中的模型进行分析可得到类似结论，该方法考虑更多客户端时所存在的动态影响。

9.5 经验教训

在包括面对面的会议或通过电话或电子邮件方式的远程协作系统会议中利用 think3 进行交互设计的阶段，think3 已获得与本章所提出的建模和验证方法相关的主要技术和工具的基本知识。实际上，已经可以以各种方式利用该形式化模型来研究 thinkteam 及其扩展系统的行为。例如包括模拟、消息序列图和模型校验器产生的反例等示例。这有助于检测 think3 在关于所提出的 thinkteam 扩展系统上进行设计时的一些不确定和模棱两可的地方。另外，think3 利用 thinkteam 的扩展模型以及发布/订阅事件通知服务功能作为实现扩展的基础，这当然可增加系统设计实用性的信心。对 think3 来说，在实际实施之前，并发系统的模型校验规范已真正地令人大开眼界。现已意识到协同系统（如 thinkteam）固有的并发特性及其复杂行为。另外，在合作过程中开发的相对简单、轻量级以及高度抽象的模型已证明在讨论更加详细的设计和实现问题之前，对于 thinkteam 接口开发的关键问题具有很大帮助。

9.6 小结

本章针对 think3 的 PDM 应用程序 thinkteam 在小型但相关的工业案例中的可用性方面进行了定性性能和定量性能的形式化分析。对具有随机进程代数 PEPA 的复制库的附加功能进行建模，并验证了在具有随机模型校验器 PRISM 的模型中以随机时态逻辑 CSL 表示的性能。验证的性能包括如库赋值、编辑权限互斥和消除死锁的功能性性能，以及特别是在连接服务质量、库可用性和用户工作模式等不同假设条件下可提供 thinkteam 可用性信息的性能。

PEPA 规范的图形化表示有助于与来自工厂的 think3 同事进行模型开发和讨论。此外，现已拥有多个独立设计部门的许多重要制造业都在使用 thinkteam，其中，各个设计部门都需要可靠且高效的软件系统来以一种高效方式进行协作。在软件产品化时应考虑 think3 中许多固有的并发性问题，以及在进行产品质量评估时所意识到的难点问题，都可通过提高在软件设计的早期阶段利用模型校验技术的兴趣来简化。

与 think3 密切合作而开发的 thinkteam 模型已表明这项工作对于获得精确而明确的规范具有重要意义，同时有助于提供有关 thinkteam 的更好文档。另一方面，通过分析实际应用 thinkteam 的日志文件而获得的信息对于模型和结果具有相当大的作用。进一步分析这些数据有助于获得用于分析 thinkteam 在不同使用模式下的模型。相信这是值得深入研究的具有重大意义的主题。反过来，这些模

型和结果也在改进记录 thinkteam 用户活动的日志中发挥作用，以获得更多有关 thinkteam 可用性方面的理解。

致谢

本章所介绍的工作是由意大利项目 TOCAI（MIUR/FIRB）和 XXL（CNR/RSTL）提供部分资助的。thinkteam、think3 和 think-PLM 为 think3 公司的注册商标。

参考文献

1. E. Clarke, E. Emerson, and A. Sistla. Automatic verification of finite-state concurrent systems using temporal logic specifications. *ACM Transactions on Programming Languages and Systems*, 8:244–263, 1986.
2. E. M. Clarke, Jr., O. Grumberg, and D. A. Peled. *Model Checking*. MIT Press, 1999.
3. M. Kwiatkowska, G. Norman, and D. Parker. Stochastic model checking. In *Formal Methods for the Design of Computer, Communication and Software Systems: Performance Evaluation, Lecture Notes in Computer Science 4486*, pp. 220–270. Springer, 2007.
4. D. Wells. Extreme Programming: A Gentle Introduction, 2006. Available at: http://www.extremeprogramming.org
5. J. Stark. *Product Lifecycle Management: 21st Century Paradigm for Product Realisation*. Springer, 2005.
6. M. H. ter Beek, M. Massink, D. Latella, S. Gnesi, A. Forghieri, and M. Sebastianis. Model checking publish/subscribe notification for *thinkteam*. In *Proceedings FMICS'04, Electronic Notes in Theoretical Computer Science 133*, pp. 275–294. Elsevier, 2005.
7. M. H. ter Beek, M. Massink, D. Latella, S. Gnesi, A. Forghieri, and M. Sebastianis. A case study on the automated verification of groupware protocols. In *Proceedings ICSE'05*, pp. 596–603. ACM Press, 2005.
8. M. H. ter Beek, M. Massink, and D. Latella. Towards model checking stochastic aspects of the *thinkteam* user interface. In *Proceedings DSVIS'05, Lecture Notes in Computer Science 3941*, pp. 39–50. Springer, 2006.
9. P. Buchholz, J. Katoen, P. Kemper, and C. Tepper. Model-checking large structured Markov chains. *Journal of Logic and Algebraic Programming*, 56:69–96, 2003.
10. H. Hermanns, J.-P. Katoen, J. Meyer-Kayser, and M. Siegle. A tool for model-checking Markov chains. *International Journal on Software Tools for Technology Transfer*, 4(2):153–172, 2003.
11. M. Kwiatkowska, G. Norman, and D. Parker. Probabilistic symbolic model checking with PRISM: A hybrid approach. In *Proceedings TACAS'02, Lecture Notes in Computer Science 2280*, pp. 52–66. Springer, 2002.
12. H. Younes and R. Simmons. Probabilistic verification of discrete event systems using acceptance sampling. In *Proceedings CAV'02, Lecture Notes in Computer Science 2404*, pp. 223–235. Springer, 2002.

13. M. Massink, D. Latella, M. H. ter Beek, M. D. Harrison, and M. Loreti. A fluid flow approach to usability analysis of multi-user systems. In *Proceedings HCSE'08, Lecture Notes in Computer Science 5247*, pp. 166–180. Springer, 2008.
14. J. Hillston. Fluid flow approximation of PEPA models. In *Proceedings QEST'05*, pp. 33–43. IEEE Press, 2005.
15. J. Hillston. *A Compositional Approach to Performance Modelling*. Cambridge University Press, 1996.
16. M. Tribastone. The PEPA Plug-In Project. In *Proceedings QEST'07*, pp. 53–54. IEEE Press, 2007.
17. M. H. ter Beek, S. Gnesi, D. Latella, M. Massink, M. Sebastianis, and G. Trentanni. Assisting the design of a groupware system—Model checking usability aspects of *thinkteam*. *Journal of Logic and Algebraic Programming*, 78:191–232, 2009.
18. R. Levesque. *SPSS Programming and Data Management*. SPSS, 2007. Available at: http://www.spss.com
19. T. Hérault, R. Lassaigne, F. Magniette, and S. Peyronnet. Approximate probabilistic model checking. In *Proceedings VMCAI'04, Lecture Notes in Computer Science 2937*, pp. 307–329. Springer, 2004.
20. D. Parker, G. Norman, and M. Kwiatkowska. *PRISM 2.0—Users' Guide*, February 2004. Available at: http://www.cs.bham.ac.uk/~dxp/prism
21. V. Kulkarni. *Modeling and Analysis of Stochastic Systems*. Chapman & Hall, 1995.

第 6 部分

运行时：测试和模型学习

第 10 章　测试和测试控制符号 TTCN-3 及其应用

Ina Dchieferdecker 和 Alain-Georges Vouffo-Feudjio
弗劳恩霍夫 FOKUS，柏林，德国

10.1　简介

为了满足具有能够描述测试行为规范的普遍理解语言句法的迫切需要，而创建了 TTCN-3 语言[1]。工业和科学上为满足所有黑箱、灰箱测试所需的一个单一测试符号促进了 TTCN-3 语言的发展。与早期的测试技术相比，TTCN-3 提倡采用常见方法和风格，以使得套件和产品的测试维护更为简单。在 TTCN-3 的作用下，测试人员可指定抽象层的测试套件并侧重于检查测试目的而非测试系统适应和执行细节的测试逻辑。标准化语言向测试套件提供者和用户提供了很多优势。而且，由于可获得大量文档、示例和预先设定的测试套件，采用标准化语言还可减少教学和训练的成本。显然，在对于不同测试类型下测试而非学习不同技术时更倾向于采用相同的语言。稳定应用以及在 TTCN-3 供应商与用户之间的合作确保了该语言的一致性维护和发展。

TTCN-3 是树和表相结合的符号（TTCN）的后续版本，TTCN 最初是在 1984 ~ 1997 年间由国际标准化组织（ISO）开发的以作为一种广泛应用于开放系统互联（OSI）的一致性测试标准和方法[2]。TTCN 现已成为一种着重于激励和多个响应测试中树形结构的以表格形式表示的相对静态测试规范语言。TTCN 由欧洲通信标准委员会（ETSI）通过增加模块化、并发测试和支持抽象语法符号（ASN. 1[4,5]）而逐步发展为 TTCN-2 和 TTCN-2 + +[3]。1998 年，ETSI 又开始开发 TTCN-3。TTCN-3 的第一个版本在 2000 年发布。自此以后，TTCN-3 不断更新和扩展。最新版本是 2010 年发布的 v4. 2. 1[4,6-17]。

TTCN-3 能够对各种测试类型进行基于规范的系统测试，如一致性测试、互操作性测试、回归测试、鲁棒性测试、性能测试和可扩展性测试。TTCN-3 是一种定义用于分布式系统黑箱测试和灰箱测试的测试程序的语言。可允许对简单的集中式测试行为和复杂的分布式测试行为在队列、选项方案和激励与响应的循环上进行一种简单有效的描述。测试系统可利用大量测试组件来并行执行测试程序。TTCN-3 语言具有完整定义的语法和操作语义的特点，可为 TTCN-3 测试规

范及其执行提供了一个准确的理解。

对于测试人员来说，指定包括测试数据、测试配置和测试行为的工作易于完成。一个测试系统能够与一个被测系统（SUT）进行同步通信或异步通信。测试系统和 SUT 之间的数据传输可通过模板进行定义，该模板能够利用功能强大的匹配机制来定义所用数据或期望数据的精确设置。同时，提供了一种判决处理机制来评估 SUT 的反应。测试数据和类型既可直接在 TTCN-3 中描述，也可从其他数据类型语言中导入。此外，还支持模块、测试案例或函数以及甚至 TTCN-3 模块的各种参数化形式，以使之能够灵活重用并适用于不同测试情况下的测试规范。选择将要执行的测试案例既可在测试运行中由用户直接控制，也可在模块控制部分中指定。

图 10.1 给出了 TTCN-3 语言的简要概述。TTCN-3 元模型定义了 TTCN-3 的多种概念及其之间的关系[18]。这些概念可以以文本[6]、表格[7]、图形[8]或其他性能表示形式表现出来。另外，TTCN-3 还支持导入在 ASN.1、接口定义语言（IDL[12,19]）和可扩展标记语言（XML[13,20]）中指定的数据类型和值。而且，也支持其他如结构化查询语言（SQL[19]）或网页服务描述语言（WSDL[21]）等数据形式[22]。

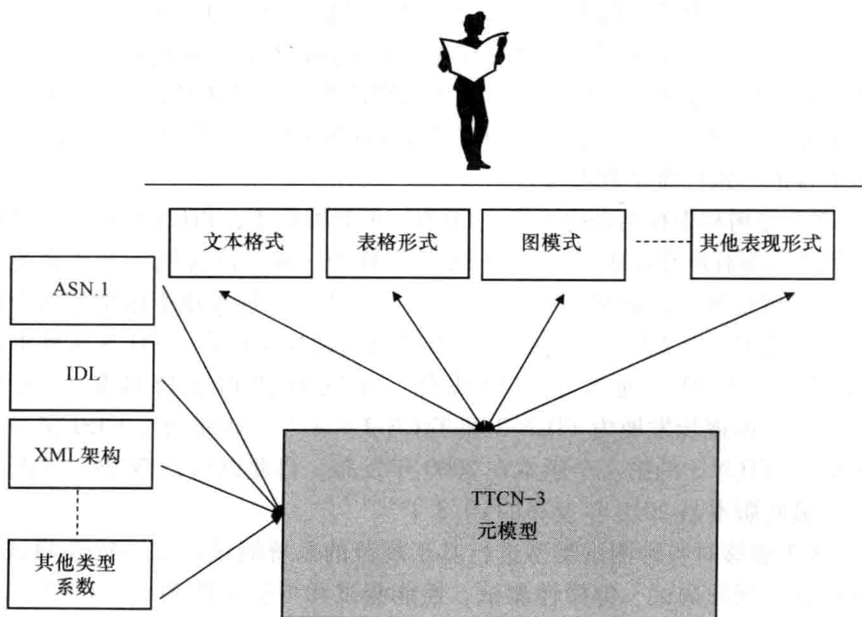


图 10.1 TTCN-3 技术的基本结构

TTCN-3 的 ETSI 标准包括了在“测试及规范方法；测试与测试控制符号第 3

版”系列文档中分为的 10 部分：

1）TTCN-3 核心语言[6]。该文档规范了 TTCN-3 语言的语法以及如何由一组 TTCN-3 模块来定义测试套件。

2）表格表示格式（TFT[7]）。TTCN-3 除了提供一种文本格式外还提供了其他表示格式。表格格式在形式和功能上与早期的 TTCN-2 版本[20]很相似。这是专为更喜欢 TTCN-2 早期版本编写测试套件风格的用户设计的。一个 TTCN-3 模块在 TFT 中表示为一组表格。然而，由于这种格式没有太多用户，并在 2007 年发布了最新版本。

3）图形表示格式（GFT[8]）。这是 TTCN-3 中第 2 种基于消息队列图（MSC[19]）的表示格式。在 GFT 中，TTCN-3 的行为定义由队列图表示。直到统一建模语言（UML）测试图[23]越来越多地用作 TTCN-3 的另一种前端，GFT 曾被广泛运用。

4）可操作性语义[9]。第 4 部分介绍了 TTCN-3 的行为结构的含义，并提供了一个面向状态的 TTCN-3 模块执行视图。

5）TTCN-3 运行时接口（TRI[10]）。一个基于 TTCN-3 的测试系统需要一个适用于 SUT 处理通信和定时问题的特殊平台。TRI 定义了一组适用于测试系统到一个 SUT 的接口，这些接口是由系统适配器实现的。

6）TTCN-3 控制接口（TCI[11]）。另外，一个基于 TTCN-3 的测试系统还需要一个适用于解决测试管理、测试组件处理、外部数据编码/解码以及日志记录的测试平台。由 TCI 定义的这个测试平台的一组接口是由测试适配器实现的。

另外，还开发了一些有关语言映射和其他语言特点的附加标准，如下所述：

7）从 ASN. 1 模块到 TTCN-3 的语言映射[4]。

8）从 IDL 规范到 TTCN-3 的语言映射[12]。

9）从 XML 结构到 TTCN-3 的语言映射[13]。

10）对 TTCN-3 模块中文档标记的支持[14]。

在 2010 年，在 TTCN-3 中增加了扩展包。一个扩展包定义了一组关于 TTCN-3 应用于特定应用领域的附加但可选的概念。例如，实时概念对于测试嵌入式或时间关键的系统非常有用，但对于普通的测试并不需要。TTCN-3 的扩展包包括：

11）支持静态配置及部署[15]。

12）支持包括如类型参数化的高级参数化[16]。

13）支持行为类型以允许如函数动态分配等[17]。

14）支持 TTCN-3 的性能和实时测试[24]。

TTCN-3 是一种不断维持随着工业用户的需求而增加的新标准的测试技术。最新的信息可在由 ETSI 维护的 TTCN-3 主页中获得[25]。

10.2 TTCN-3 概念

TTCN-3 的核心语言是一种测试规范和测试实现语言，具有现代编程语言的形式和内涵。除了典型的编程结构外，TTCN-3 还包含了所有不同测试形式下规范测试数据和测试程序所需的重要特征，其中包括测试判决、数据模板、定时器处理、测试组件、同步和异步通信、记录日志和类似这些特征。TTCN-3 测试规范能够从其他模块中输入定义；定义类型和测试数据；规范函数、altsteps 和测试案例中的测试行为，并定义控制部分中测试案例的选择、参数化和执行（见图 10.2）。

TTCN-3模块	
输入	利用其他模块的定义
数据类型	用户自定义数据类型
模块	发送或沿测试接收测试的数据
测试配置	测试组件类型和端口类型
测试行为	功能、altsteps和测试案例
测试控制	测试案例的选择、参数化和执行

图 10.2 TTCN-3 的模块结构

10.2.1 模块

一种基于 TTCN-3 的测试规范中的高层构建块是 module。一个模块中包含了除了其他 TTCN-3 的结构，但不包括子模块。模块可输入其他模块的完全定义或部分定义。模块可由运行时测试环境提供的值来参数化。对于参数，还可指定默认值。

一个 TTCN-3 模块由两部分组成：模块定义部分和模块控制部分。定义部分包含了由模块定义的数据（函数、测试案例、组件、类型、模板），这些数据都能在整个模块内部使用，也可从其他模块输入。控制部分是一个模块的主要程序，描述了测试案例或函数的执行顺序。模块可访问测试案例输出的判决，并根

据这些判决来决定测试执行的下一步动作。TTCN-3 中的测试行为在函数、altsteps 和测试案例中进行定义。模块的控制部分可调用任何在其属于的模块中定义或输入的测试案例或函数。

10.2.2　测试系统

一个测试案例是通过测试系统来执行的。TTCN-3 允许规范静态集中式或动态并发式测试系统。一个测试系统由具有完整定义的通信端口的一组互联测试组件和一个用于定义测试系统与 SUT 边界的显式测试系统接口组成。

在每个测试系统中，都有一个主测试组件（MTC）。而其余所有的其他组件都称为并行测试组件（PTC）。在每个测试案例开始执行时，自动创建并执行 MCT。在 MTC 执行结束时，测试案例也结束，也意味着结束了所有其他 PTC 的执行。MTC 的行为是在测试案例本体中指定的。在一个测试案例执行期间，PTC 可动态地创建、开始和停止。一个测试组件可自动停止，也可由其他测试组件停止。

为了通信的目的，每个测试组件都有一组局部端口。每个端口都有输入方向和输出方向。输入方向建模为一种先入先出（FIFO）的无限队列，其中存储着输入信息，直到这些信息通过测试组件自身端口进行处理。输出方向直接与通信部分连接成为另一个测试组件或 SUT，即输出信息没有缓存。

在测试执行期间，TTCN-3 可区分连接端口和映射端口。连接接口用来与其他测试组件进行通信。如果两个端口相连，则一个端口的输入方向与另一个端口的输出方向相连，反之亦然。映射接口是用来与 SUT 进行通信。将一个测试组件的自身端口映射到抽象测试系统接口的一个端口可看作定义信息流如何被引用的一种单纯的名字迁移。TTCN-3 可区分抽象测试系统接口和实际测试系统接口。抽象测试系统接口建模为一个定义 SUT 抽象接口的端口集合。而实际测试系统接口是一种基于 TTCN-3 的测试环境的特殊应用。可实现 SUT 的实际接口，且在 TRI 中定义（TTCN-3 的第 5 部分）。

在 TTCN-3 中，在运行时可动态创建和释放连接和映射。一个组件的连接和映射个数在逻辑上没有限制，但仅限制了所用的测试设备个数。一个组件（甚至一个端口）可以与自身相连。同时，也允许一对多的连接。在测试执行期间通信双方需唯一指定的情况下可采用一对一的通信，而通过组播和广播方式可采用一对多的通信。对于测试组件之间以及测试组件与 SUT 之间的通信，TTCN-3 支持基于消息和基于过程的通信。基于消息的通信是根据一种发送方和接收方相互独立进行的异步消息交换（除了只能在发送后才能接收的消息之外）。基于过程的通信主要以远程实体中的同步调用程序为基础。在基于过程的通信中，原则上会阻断调用组件，直到接收到一个调用响应或异常。在 TTCN-3 中，出于测试目的都可指定调用程序并等待响应或异常的客户端，以及接受调用并输出响应或

产生异常的服务器端。

10.2.3 测试案例和测试判决

测试案例定义了用于执行检查 SUT 是否通过测试的测试行为。与模块类似，认为测试案例是一个用于检查给定测试目的测试过程的独立且完整的规范。测试案例的执行结果即测试判决书。

TTCN-3 为解释测试运行提供了一种特殊的测试判决机制。该机制是通过一组预先设定的判决、局部测试和全局测试判决以及读取和设置局部测试判决的操作来实现的。预先设定的判决包括 pass、inconc、fail、error 和 none。这些判决都是用来判断测试完整运行或局部运行的。pass 判决表示 SUT 的行为是根据测试目的的；fail 判决表示 SUT 违反了其规范；inconc（不确定）判决描述了一种既非 pass 也非 fail 的情况；none 判决是局部和全局测试判决的初始值，即还没有赋值任何判决。在测试执行期间，每个测试组件都保持其自身的局部测试判决。一个局部测试判决是一个在组件创建时每个测试组件的自动实例化对象。一个测试组件能够检索并设置其局部判决。error 判决是指不允许由测试组件来进行设置。如果在测试设备中产生一个错误，则通过 TTCN-3 的运行时环境自动设置。当一个局部测试判决值发生变化时，应运用特殊的重写规则。重写规则只可能使得测试判决变得更差；如 pass 判决可能变为 inconc 判决或 fail 判决，但 fail 判决却不能变为 pass 判决或 inconc 判决。

除了局部测试判决，TTCN-3 运行时环境还对每个测试案例都保持一个全局测试判决。该全局测试判决不能由测试组件访问。当测试组件结束测试时，可根据重写规则进行更新。当测试案例结束时，在模块控制部分会返回最终的全局测试判决。

10.2.4 备选方案和快照

TTCN-3 语义的一个特殊功能是快照。快照与测试组件的行为有关。对于由于发生超时，测试组件终止运行，以及信息接收、程序调用、程序回应或其他异常行为等原因产生的行为分支，快照是十分必要的。在 TTCN-3 中，行为分支是通过 alt 声明语句来定义的。

一个 alt 声明语句描述了备选方案的有序集合，即一个行为的选项方案分支的有序集合。每个备选方案都有一个守护条件。一个守护条件中包括几个可能是指变量值的前置条件、定时器状态、端口队列容量以及组件、端口和定时器的识别码。相同的前置条件可用于不同的守护条件。如果满足相应的守护条件，即可执行备选方案。如果有多个备选方案都可执行，则执行备选方案列表中的第一个可执行备选方案。如果没有可执行的备选方案，则再次执行 alt 声明语句。

评估多个守护条件需要一段时间。在这段时间内，前置条件可能会动态变

化。如果一个前置条件在不同的守护条件下多次验证，将会导致守护条件评估的不一致。TTCN-3 通过利用快照来避免产生上述问题。快照是包括 alt 声明语句评估所需所有信息的局部模块状态。在执行一个备选方案时，会选择（即记录）一个快照。对于前置条件的验证，只需利用当前快照的信息。因此，前置条件的动态变化不会影响守护条件的评估。

10.2.5　默认处理

在 TTCN-3 中，默认值是用于备选方案的重复设置，这经常用于处理一些非期望响应或无响应。默认行为可通过 altsteps 来指定，并作为默认值。对于每个测试组件，默认行为（即激活的 altsteps）都保存在激活顺序的默认列表中。TTCN-3 的操作激活或禁用这些默认列表中的操作。激活操作会在列表末端增加新的默认行为，而禁用操作会从列表中删除默认行为。

如果当前快照并非 alt 匹配中的任何一个备选方案，则在每个 alt 声明语句之后会调用激活的默认行为。正如一个 alt 中的选项方案，激活默认行为及其选项方案会逐个进行评估。如果某个选项方案匹配，则执行该激活的默认行为。如果没有一个激活默认行为的选项方案匹配，则进行新的快照，并再次执行 alt 声明语句。

在默认行为成功匹配的情况下，则默认行为会通过一个 stop 声明语句来停止执行测试组件，这可能会 repeat 再次执行 alt 声明语句，或在调用默认行为的 alt 声明语句之后立刻继续执行测试组件的主控制流。

TTCN-3 使用 altstep 来指定默认行为或结构化选项方案的集合。altsteps 的语义与选项方案和快照密切相关。正如在一个 alt 声明语句中，一个 altstep 定义了选项方案的有序集合。不同之处在于在执行一个 altstep 时不会进行快照。altstep 选项方案的评估是基于从调用的 alt 声明语句中进行的当前快照。从概念上讲，altstep 的选项方案增加到 alt 声明语句的选项方案集合中。另外，也可以像调用函数一样来调用 altstep。在这种情况下，可将 altstep 看作一个只调用 altstep 的 alt 声明语句。也就是说，altstep 的选项方案仅是该 alt 声明语句中的选项方案。

10.2.6　通信操作

对于规范测试行为，通信操作非常重要。TTCN-3 支持基于消息和基于过程的通信。通信操作可分为两部分：向 SUT 发送信息的激励和用于描述 SUT 反应的响应。

在 TTCN-3 中，基于消息的通信是异步通信，这意味着消息的发送是没有阻塞的；在消息发送之后，系统不必等待响应。接收操作会阻断执行，直到在指定端口接收到一条与之匹配的消息。接收操作定义了一个用于接收有效消息的匹配部分。另外，也可指定一个地址来确定来自期望消息的连接。定义以下操作来实

现基于消息的通信：

- send，直接向 SUT 或另一个 PTC 发送消息；
- receive，从 SUR 或另一个 PTC 接收消息；
- trigger，忽略所有输入消息，直到期望消息到达。

在 TTCN-3 中，基于过程的通信为同步通信，也就是说消息的调用基本上会被阻断。对于同步通信，定义以下操作：

- call，调用一个远程程序；
- getcall，接受调用；
- reply，对调用作出回应；
- getreply，接受调用的回应；
- raise，提供一个异常调用；
- catch，接受一个异常调用。

10.2.7 测试数据规范

一个测试系统需与 SUT 交换数据并在 PTC 之间交换数据。与 SUT 的通信可以是通过从/到 SUT 发送/接收消息的异步形式，也可以是通过调用 SUT 程序或从 SUT 接收调用程序的同步形式。在这些情况下，应根据 SUT 规范由测试系统来定义测试数据。TTCN-3 可提供不同结构来描述测试数据：类型、模板、变量和程序签名等。这些测试数据都能用来表示协议消息、服务原语、程序调用、异常处理。除此以外，TTCN-3 还可能会导入如 ASN. 1、IDL 或 XML 等其他语言描述的数据。

在可指定用户定义类型的基础上，TTCN-3 还提供了一系列基本类型和预定义类型。大部分类型与所熟悉的程序语言中的类型相似，如 Java 或 C。但其中一些类型是专用于测试领域的：

- port 类型，用于定义通过 PTC 之间或与 SUT 之间进行通信的一个给定端口类型的端口（实例）进行通信的消息或程序类型。
- component 类型，用于定义本地端口、变量、常量和一个给定组件类型的 MTC 或 PTC 的定时器。
- verdict type，是一种所有可能作为测试案例结果的判决枚举类型。
- any type，是所有 TTCN-3 类型的联合类型，用于在已知具体数据时指定应评估的通用数据。
- default 类型，用于处理默认选项的类型。

TTCN-3 还支持有序的结构类型，如 record、record of 、set、set of、enumerated 和 union。对于基于过程的通信、程序的签名也可定义。签名的特点是具有名称、可选参数列表、可选返回值和可选异常列表。

一个表示给定类型的测试数据的 template 用于定义所发送或接收的消息。模

板定义了要传输的不同值或一组评估接收消息是否匹配的值（通过使用通配符或其他匹配机制）。将接收消息与模板进行比较，如果消息为模块所定义的一组值的某个值，则模板匹配。对于基本类型、用户定义类型或程序签名，可指定模板。在其他模板定义中，这些类型可参数化、重复使用或修改。

在此没有给出 TTCN-3 的所有细节。想要进一步了解该技术，请参见标准化丛书[4,6-17]、其他论文[25-27]和工具书[22]。

10.3 入门示例

接下来，为深入了解该技术，在此给出一个 TTCN-3 中小测试规范。即用于三角形分类的由 G. J. Myers[28]提出的著名三角形测试示例。根据三角形的边长可分为不等边三角形、等腰三角形和等边三角形。另外，用于定义三角形的数据可能是错误的，或即使数据正确但不是用于定义三角形。

当测试三角形分类的软件时，必须提出 SUT 的类型问题。在 TTCN-3 中，这种情况下可通过定义 Triangle 以组成 3 个不受限制的 integer 值和通过 Classification 来指定可能的三角形类型来明确定义 SUT 的类型：

```
type integer Triangle [3] (0.. infinity);
//定义一个三角形的所有个数
type ennumerated Classification{
//三角形的属性
syntacticallyIncorrect,
noTriangle,
scaleneTriangle,
isoscelesTriangle,
equiateralTriangle
}
```

接下来，根据在输入方向发送 Triangle 值和在输出方向发送 classification 值的基于消息的端口可定义 SUT 的接口 DetermineTriangle。SUT 的测试组件采用由测试组件类型 TriangleleTester 定义的端口：

```
type port DetermineTriangle message
{out Triangle;in Classification}
//SUT 的接口
type component TriangleTester
{port DetermineTriangle b}
//测试组件
```

这些定义已足以将 SUT 的具体测试定义为由检查是否为一个具体等边三角形的测试案例 Simple 来阐述：

```
testcase Simple() runs on TriangleTester{
//一个简单测试案例
b.send(Triangle:{2,2,2});
//所要检查的三角形
b.receive(Classification:equilateralTriangle);
//期望响应
setverdict(pass);//成功测试
}
```

上述给定的测试规范是将测试数据和测试行为紧密相连，但这样既没有太多用处也不易维护。另一种方法是将从测试行为中分离出测试数据。为此，又定义了一个包含三角形 3 个整数值及其分类的类型 TestVector。然后再以 vaild1、valid2 等模板形式来定义具体的三角形 3 个整数值以及分类。

```
type record TestVector{
//输入和期望输出的结合
Triangle input,
  Classification output
}
template TestVector valid1:={{2,2,2},
isoscelesTriangle};
//等
template TestVector invalid1:={{1,0,3},noTriangle};
//无效的三角形
template TestVector invalid2:={{1,2,3},noTriangle};
//等
```

另外，还需定义一个如何评估 SUT 非期望响应或无响应的测试规范。通过接收任何语句（alt 语句中第 2 个选项）来处理非期望响应，只有在选择了正确响应（alt 语句中第 1 个选项）之后才作出评估。为确定 SUT 无响应，只要 SUT 接收到一个三角形分类队列，则一个本地定时器 noResponse 被定义、启动并观察是否超时：

```
testcase Extended(TestVector tv) runs on TriangleTester
{
//参数化测试
timer noResponse:=1.0;
```

```
    //定时器超时 1 秒
    b.send(tv.input);noRespones.start;
    alt{
    [ ]b.receive(tv.output){setverdict(pass);}
    //期望的正确响应
      [ ]b.receive{setverdict(fail);}
  //非期望的错误响应
  [ ]noResponse.timeout{setverdict(fail);}
  //完全没有响应
  }
  }
```

可对该示例进一步扩展，例如，采用参数化使得测试数据更加灵活或通过 SUT 的两个并行测试组件队列来检查是否可并发使用。

通过利用 TTCN-3 工具[22]，可开发、实现并有效执行用于三角形定义示例的 TTCN-3 测试规范（见图 10.3）。

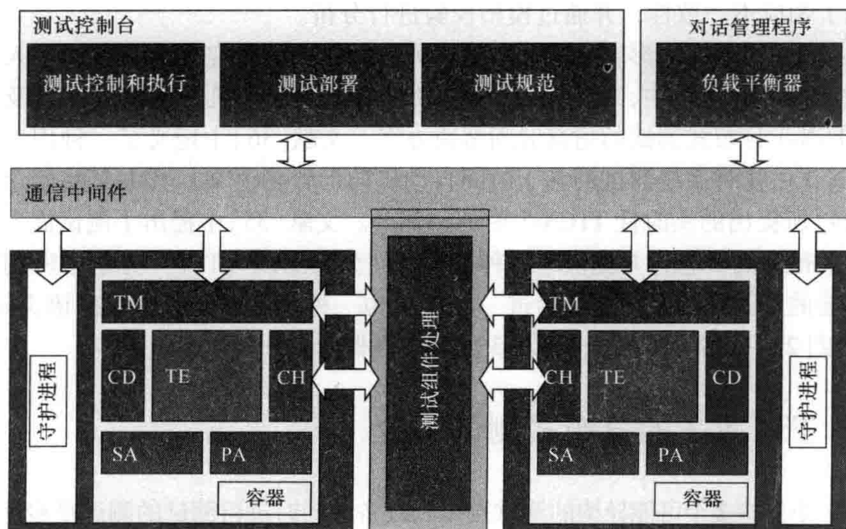


图 10.3 基于 Eclipse 的一个 TTCN-3 集成开发环境

10.4 TTCN-3 语义及其应用

TTCN-3 的语义是由一个操作语义[9]定义的，以一种直观而明确的方式来定义 TTCN-3 行为的含义。这本身并不是一种形式化语义，因此形式化检验和验证

的能力是有限的。事实上，TTCN-3 的操作语义对 TTCN-3 模块的执行提供了一种面向状态的视图。该视图仅定义了 TTCN-3 中行为的含义，即功能、可选步长、测试案例、模块控制和如通信操作或控制流语句等行为描述语句。

操作语义利用了所要执行的 TTCN-3 模块的控制流图。控制流图是一种由描述以控制流图表示的行为执行过程中可能存在的控制流的标记节点和标记边组成的有向图。操作语义用来构建运行环境（见下节）、测试解决方案（见文献[59]），和为进一步检验和验证所需的形式化语义定义提供基础。

分析和提高 TTCN-3 测试套件的可靠性是一个主要的研究方向。因此，应对 TTCN-3 的语义进行约束扩展，以实现在 TTCN-3 测试套件中数据和行为更为精确。在文献[30]中简要概述了出于维护目的的优化测试质量，其中定义了构成确定代码好坏并重构测试套件评估和自动调整基础的度量、模式和反模式[30]。在文献[32]中提出了完善测试套件设计、维护和自动验证的使用准则。文献[33]中提出了一种分析和优化基于 TTCN-3 的测试套件中测试数据范围的方法。

此外，还开展了验证 TTCN-3 中测试行为的研究。特别是，已能够识别如在测试案例中和测试案例之间测试行为不一致的行为异常。例如，在文献[34]中已明确了响应不一致性，并通过模型校验进行分析。

在基于 TTCN-3 的形式化测试方面，另一个主要的研究方向是测试嵌入式实时系统：在文献[35]中，提供了一种仿真时间下基于主机的测试语义以及一种用于 TTCN-3 分布式测试的仿真时间解决方案。文献[36]中定义了一种用于扩展的 TTCN-3 中实时系统测试的基于时间自动机的形式化语义。其中某些概念用于文献[24]所提出的实时性 TTCN-3 中的一部分。文献[35]中提出了测试嵌入式系统中连续行为的概念，并提供了一种基于时间分区测试（TPT）方法中所用的流处理功能的形式化语义。这随后进一步扩展为一种基于混合自动机的语义定义，且在文献[24]中增加了用于 TTCN-3 的实时性概念。

10.5 TTCN-3 的分布式测试平台

在一个支持多个可能异构的测试接口和设备上同时进行测试的测试平台上可实现 TTCN-3 的所有潜能。通过该测试平台，不仅可以在使用互联测试设备的测试系统中了解 SUT 的所有可能分布式和异构配置，而且还能解决测试系统的性能和可扩展性的问题。

现已利用一个公共对象请求代理体系架构（CORBA[37]）插件开发了一个 TTCN-3 的分布式测试平台。利用该平台上不仅可以进行性能、可扩展性和负载以及压力的测试，还可以进行非分布式功能测试或只需进行单个设备和/或接口的测试。在此，并不详细描述所有实现细节，而只是简要概述其中最主要的方面（见图 10.4）。

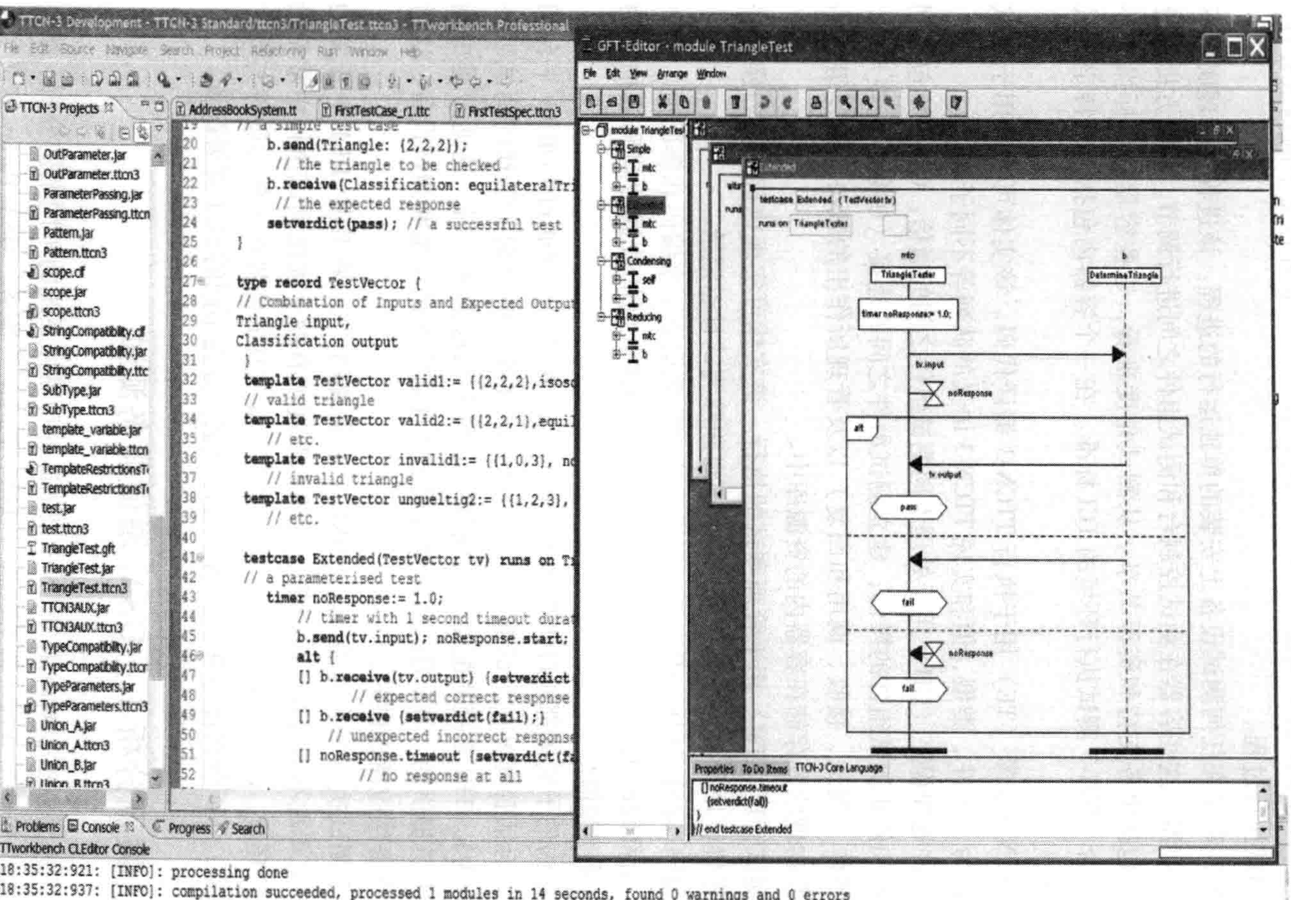


图 10.4　TTCN-3 的分布式测试平台架构

测试控制台是该平台的控制核心，同时也是一个集成开发环境（IDE），可为特定的 TTCN-3 测试案例提供支持，创建测试会话，部署容器中的测试套件，并控制测试执行过程。

守护进程是在任何测试设备上安装的单独运行的进程。该进程可管理属于不同会话的容器。这些容器在测试控制台和测试组件之间进行调节，为双方提供透明服务，包括事务支持和资源共享。从测试角度来看，这些容器是目标操作环境，且符合 TTCN-3 测试执行环境的 TCI 标准。在一个容器内包括以下特定测试系统实体：

- TTCN-3 引擎（TE）用于执行 TTCN-3 编译代码。该实体可管理用于测试控制、行为、组件、类型、值和实现 TTCN-3 语义的队列等不同子实体。
- 组件处理程序（CH）用于处理不同测试组件之间的通信。CH 接口包括创建、开始和停止测试组件的操作；建立测试组件之间的连接（映射、链接）；处理通信操作（发送、接收、调用和回复）以及管理所作出的裁决。创建组件的信息及其物理位置存储在容器内的资源库中。
- 测试管理（TM）用于管理测试的执行。该实体可实现执行测试的操作并提供和设置模块参数和外部常量，同时还可跟踪测试日志记录。
- 编码/解码（CD）用于根据假设或给定的数据类型来编码和解码值。每个 TTCN-3 值都编码为字符串与 SUT 进行通信，而接收到的数据被解码成抽象的 TTCN-3 类型和值。

容器内的子实体都通过 TRI 和 TCI 进行功能界定，同时通过 CORBA 平台相互通信。会话管理可在测试部署（而不是测试规范）中协调所选择的测试组件（及其行为）。TRI 系统适配器可实现与特定 SUT 的通信和定时，这通常是由测试开发人员提供的。TCI 测试适配器可用于实现适应测试环境和测试设备，这通常是由设备供应商提供的。

整个测试执行平台是在 Java JDK1.4 中实现的[38]。采用由 JDK1.4 开发的 CORBA 来实现不同容器间的通信。按照标准规范有助于优化测试执行中常见操作的执行环境。在 TTCN-3 平台上采用完全 Java 实现的优势在于使得硬件与操作系统无关，从而使得能够在不同场景进行测试，并为不同的 SUT 部署配置。

10.6 案例分析 I：OSA/增值服务测试

接下来，介绍表明应用 TTCN-3 来测试各种目标系统的不同案例分析。第一个案例是关于测试开放式服务架构（OSA）/Parlay[39]服务。OSA/Parlay 定义了一系列由第 3 代移动系统合作项目（3GPP）、ETSI 和 Parlay 小组联合开发的标准应用程序接口（AIP）来方便地创建通信服务。该 AIP 为现有网络提供一个通用接

口，以支持软件开发人员来开发电信应用程序。

在客户端和服务器之间的通信接口明确指定情况下，即给定如 IDL 或 WSDL 的接口定义语言，TTCN-3 尤其适合于测试客户端—服务器应用程序。基于这些规范，可产生服务器向其客户端提供的操作的功能测试以及利用服务器的客户端应用程序的功能测试。

在图 10.5 中描述了 OSA/Parlay 的架构。OSA/Parlay 定义了 3 种不同类型的组件：应用程序、提供服务功能的服务器以及框架：

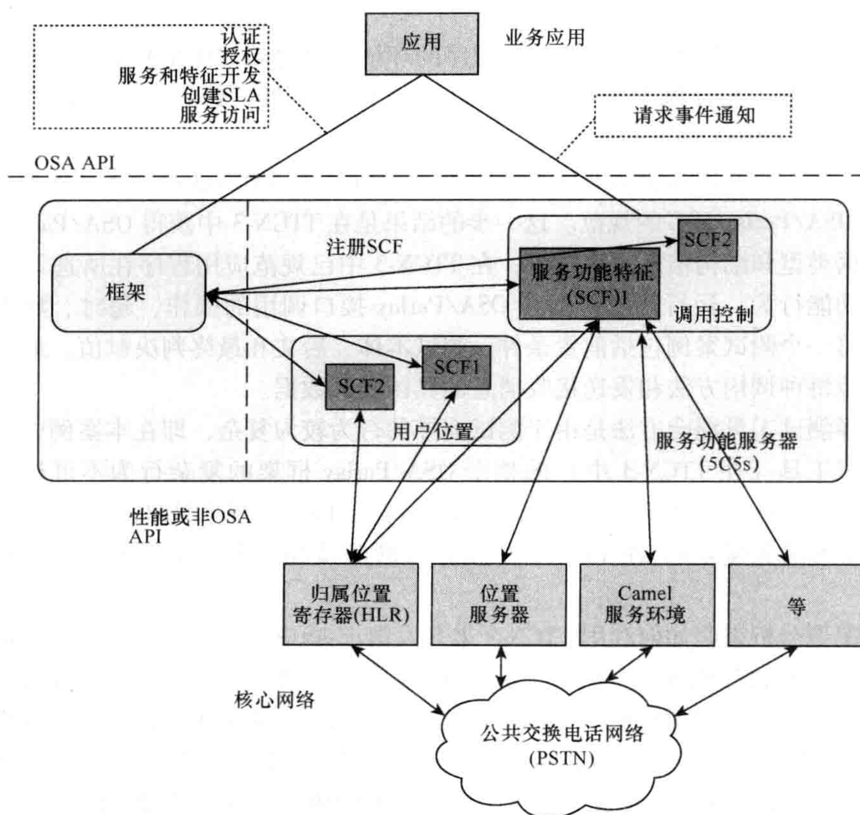


图 10.5　OSA/增值服务体系架构

- *应用程序*表示利用核心网络的任何业务应用程序。应用程序可连接和认证框架。在成功认证后，应用程序可利用该框架进行服务搜索。
- 网络提供的以及应用程序请求的*服务*称为服务功能特性（SCF）。应用程序可通过调用接口来使用一个在 OSA 技术规范中描述的 SCF。
- *框架*可为应用程序提供服务目录和连接工厂，以及 SCF 和应用程序之间的媒介。应用程序所需了解的只是工厂的位置以及需要哪些功能。

本案例分析的目的是实现一个基于 TTCN-3 的测试框架来测试基于 OSA/Parlay 的服务和应用程序。TTCN-3 用于规范框架操作、服务和/或 API 应用的测试。测试系统具有以下两个不同的作用：

- 通过 OSA/Parlay API 与框架或服务通信的服务消费者的作用。
- 框架或验证应用程序行为的服务的作用。

本案例分析中测试的应用程序只是利用 OSA/Parlay 来建立连接到网络的两个用户之间呼叫的一个小应用程序。测试系统模拟用于建立两个用户之间呼叫的 SCF 行为。

测试 TTCN-3 中 OSA/Parlay 应用程序的环境工具是将 TTCN-3 应用到由测试技术提供的 Java 编译器 TT3[22] 和由 Fraunhofer FOKUS 提供的一个 IDL-to-TTCN-3 插件和一个 CORBA 测试适配器[40]。

第一步，在利用基于标准化映射规则[12]的 IDL-to-TTCN-3 插件的 TTCN-3 中已编译 OSA/Parlay IDL 的规范。这一步的结果是在 TTCN-3 中获得 OSA/Parlay 接口的数据类型和结构信息。第二步，在 TTCN-3 中已规范应用程序在所选运行环境中的功能行为。运行环境包括在 OSA/Parlay 接口调用的操作、超时、默认行为等。每一个测试案例包括前置条件、测试本体、后文和最终判决赋值。最后一步，规范每种调用方法和发送接收消息的具体测试数据。

选择测试工具混合方法是由于测试前导的行为较为复杂，即在本案例中模拟包含测试工具（在 TTCN-3 中）的整个 OSA/Parlay 框架的复杂行为不可行。因此在测试工具链中集成了一种 OSA/Parlay 框架的 Java 实现来处理测试前导（尤其是认证和服务搜索）。在 TTCN-3 中实现反映测试场景中主要问题的实际测试本体。

本案例分析表明如何利用 TTCN-3 来开发请求-响应场景下利用同步通信机制的系统和服务的自动测试。此外，还表明语言映射（本例中为 IDL）为有效生成与接口相关的测试结构提供了一种有效方式。最后但并非不重要，本案例分析还表明不仅 TTCN-3 的解决方案非常实用，而且 TTCN-3 的开放系统架构还可以很容易地与非 TTCN-3 组件集成。进一步对案例进行研究发现测试模式的定义经常应用于通信服务基础设施的测试方案中[35]。

10.7　案例分析Ⅱ：IMS 装置测试

本案例分析的目的是将 TTCN-3 应用于定义 IMS[41] 设备的性能测试。IP 多媒体子系统（IMS）是一个用于移动和固定互联网协议（IP）服务的、开放的、标准化的、操作友好的多媒体网络架构。通过一个统一接口，IMS 可支持为无线或有线用户设备上的终端用户提供一系列丰富的服务。

图 10.6 中给出了 IMS 的参考架构。IMS 定义了用户设备（UE）和代理呼叫/会话控制功能（P-CSCF）之间的信号流量是通过一个对应于 UE 的 IPSec 通道（即通过安全 IP）来实现的。P-CSCF 是一种 IMS 终端第一个触点的会话启动协议代理（SIP 为一种如控制网络中多媒体会话[42] 的协议）。注册过程中该协议代理被分配到 IMS 终端，同时在注册期间不会变化。询问呼叫/会话控制功能（I-CSCF）可通过查询统一归属用户服务器（HSS）来检索用户位置，然后将 SIP 请求路由到其分配的服务呼叫/会话控制功能（S-CSCF）。S-CSCF 是信号平面的中心节点，决定了 SIP 信息将被转发到哪个应用程序服务器以提供相应服务。

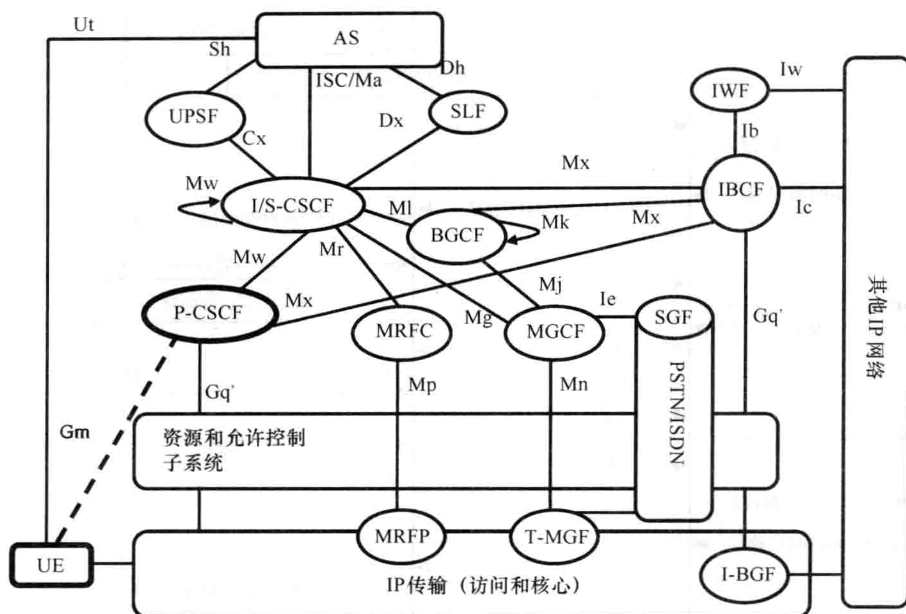


图 10.6　IMS 参考架构（来自 3GPP）

该案例分析的目的是在定义了根据精确业务类型和内容到 SUT 的输入激励信号、业务类型的统计分布、到达率以及其他相关参数的 TTCN-3 中开发 IMS 的性能测试。这些测试指定了提供给 SUT 的流量，定义了所需采取的措施，以及如何提交结果。

性能测试可测量 IMS 系统控制平面的能力。这包括相互之间执行基于 SIP 的信号传输组件以及这些组件访问的数据库。

呼叫是指由 IMS 系统来标记和管理的一个或多个用户之间的关系。用户利用 IMS 的意图是进行调用，所以在 IMS 测试中获得实际行为最显而易见的方式是对呼叫行为建模。在此对于一个传统的音频或多媒体会话采用一种可对所建立呼叫行为进行描述的语音呼叫模型。图 10.7 给出了语音呼叫所产生的消息队列。

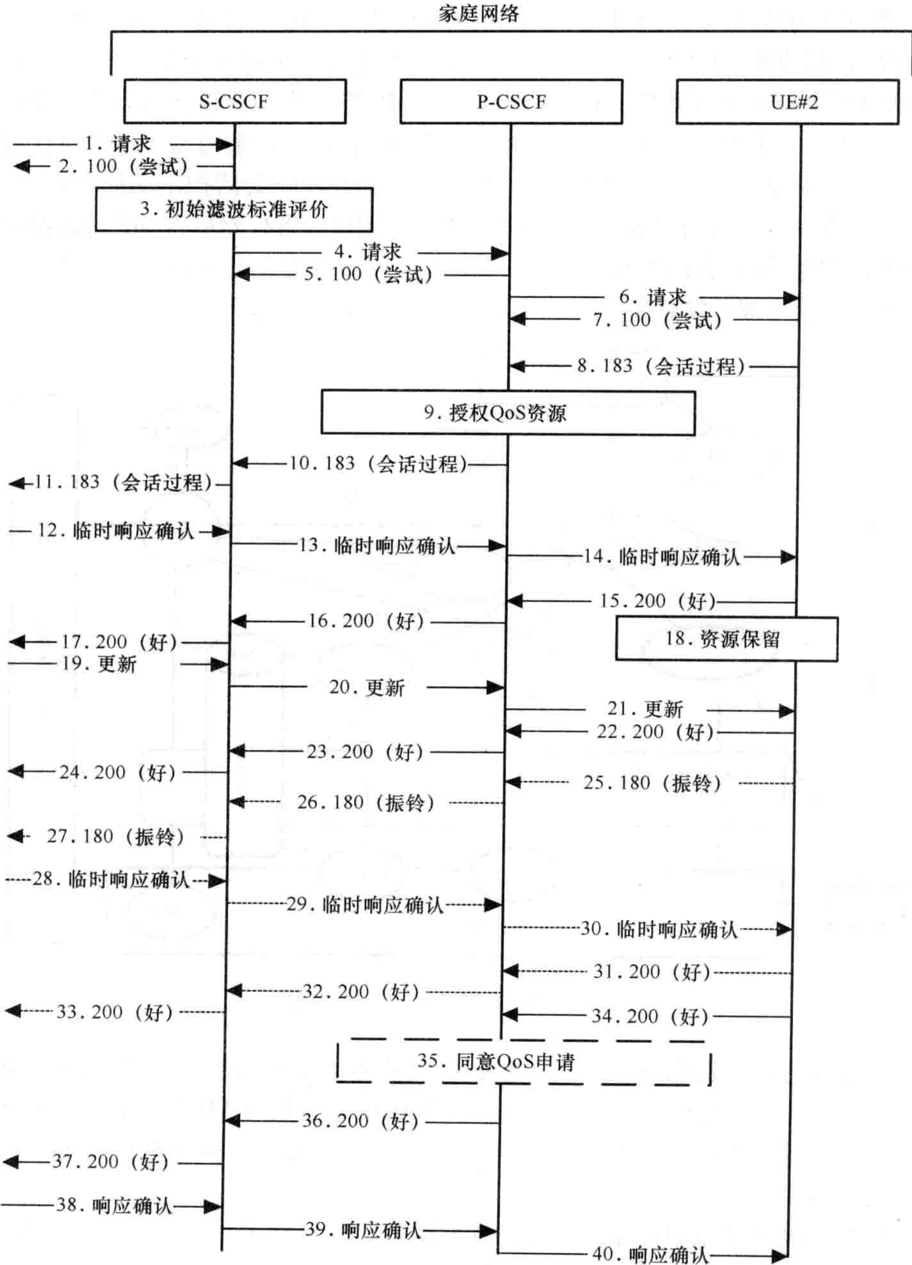


图 10.7 IMS 调用序列图

呼叫场景的是在 ETSI[43] 的 SIP 一致性测试套件和 IETF[42] 的 SIP 测试规范基础上实现的。系统类型和一致性测试的模板定义在创建 IMS 系统输入的性能测

试规范中重用。

呼叫场景采用了两个建立呼叫会话的 UE：其中一个通过将 INVITE 消息发送到 IMS 网络而起到 CALLER 的作用；另一个通过接受呼叫请求并相应响应而起到 CALLEE 的作用。

性能测试的规模取决于呼叫建立过程的并行运行。在 TTCN-3 中，UE（呼叫者和被呼叫者）作为通过系统组件与 SUT 交互的 PTC 而实现。在本案例分析中，实现两种负载测试的技术，其中，在 SUT 处于负载不断增大的情况下测量和分析其响应：

1）*应用 PTC* 。该技术是一种在 TTCN-3 中实现并行行为的显而易见的方法。呼叫者和被呼叫者同时作为 PTC 且并行执行。遗憾的是，这种技术也存在缺点，即过多测试组件的创建会占用太多的硬件资源（内存、CPU），从而可能会导致测试系统过载。

2）呼叫建立后测试组件重用。这种技术通常与第一种技术配合使用，表明一旦测试组件完成呼叫建立过程，则可重用于下一次呼叫（当然，具有新的标识符和数据集）。因此，只有一个测试组件即可在 1s 内建立多次呼叫，从而会增大总工作量。

在案例分析中评估的关键性能参数如下：

- 每秒发送的邀请个数（INVITE 消息）；
- 负载情况下测量 SUT 时的反应延迟。

图 10.8 中给出了 80s 时间内的每秒 INVITES 个数。

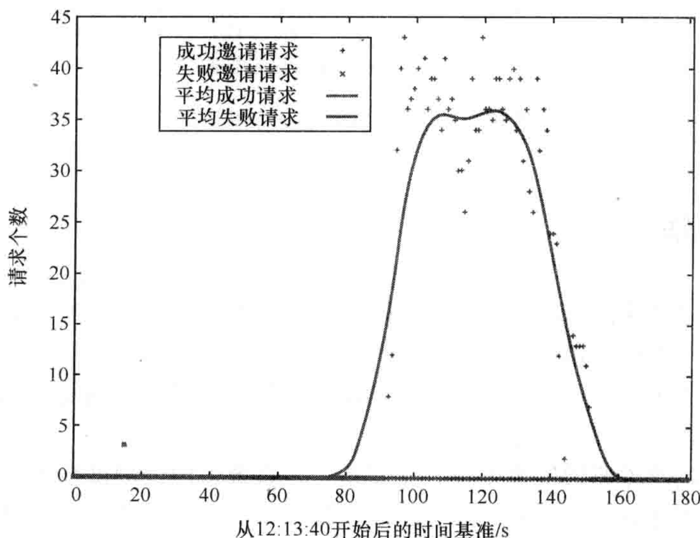


图 10.8　每秒请求个数

图 10.9 中给出了在每秒 35 INVITES 的负载情况下，SUT 反应延迟的变化。通过比较这两个图，可确定 IMS 系统内部计算需更多时间。

该案例分析表明 TTCN-3 的规范和执行功能不仅可用于功能测试，还可用于性能和负载测试。利用 TTCN-3 的功能通过可对不同测试行为赋值的 PTC 来规范分布式和并发测试的设置。此外，本案例分析还表明 TTCN-3 具有文献[11]中定义的记录格式的功能，以使得测试报告中包含基于图的测试执行评估和 SUT 定时。

对本案例进一步研究可得到由硬件供应商提供的对 IMS 具体设备的评估。在文献［44］中给出了更多结果。

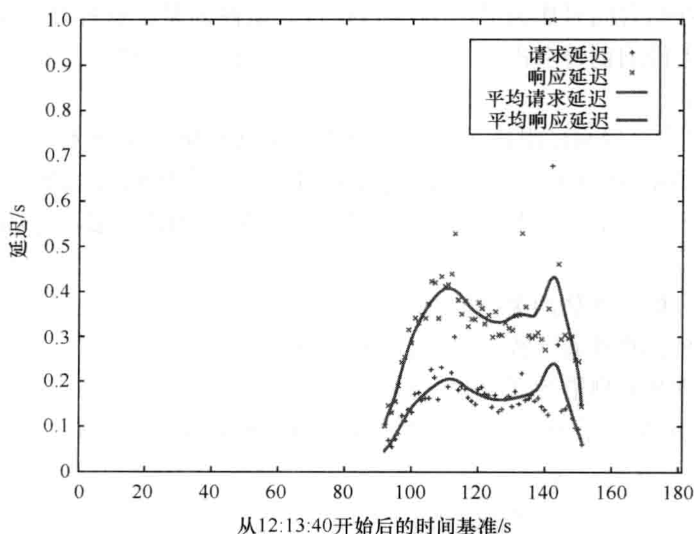


图 10.9 下载时 IMS 系统反应延迟

10.8 小结

TTCN-3 是适用于范围广泛的各种测试系统的唯一标准化测试规范和实现技术。在过去几年中，TTCN-3 取得了很大发展：现已有大量支持不同规模 TTCN-3 的工具。基于 TTCN-3 可产生各种测试解决方案。同时还计划将 TTCN-3 应用于如科学计算或云计算等其他领域。

然而，仍然需要做很多工作。尤其是，为有效采用该测试技术，需要对用户进行培训。所建立的 TTCN-3 认证证书[45]提供了一个良好的基础，但是这需要提供更多的阅读和培训材料以及培训课程。

另外，即使有免费的帮助工具，但在大学中很少有 TTCN-3 的相关课程。尤

其是，还不足以传播 TTCN-3 的相关知识及其成功案例，但还是应提供一个包括 TTCN-3 应用准则和最佳实践的全面方法。

为进一步采用 TTCN-3，需要一个坚实的基础来不断维护和发展 TTCN-3。尽管 TTCN-3 已能支持一系列非常详尽而功能强大的测试概念，但仍有很多要求，例如在与 SUT 系统交互时，必须支持面对对象的数据类型。然而，和过去一样，在被添加到核心语言和执行接口或一个 TTCN-3 的扩展包（如果不是一个新的执行接口）之前，每个新的概念都必须经过严格的审查和讨论。

参考文献

1. J. Grabowski, D. Hogrefe, G. Rethy, I. Schieferdecker, A. Wiles, and C. Willcock. An introduction into the testing and test control notation (TTCN-3). *Computer Networks Journal*, 42(3):375–403, 2003.
2. ISO/IEC 9646-3. Information Technology—Open Systems Interconnection—Conformance Testing Methodology and Framework—Part 3: The Tree and Tabular Combined Notation (TTCN), 1998.
3. ETSI TR 101 666. Information Technology Open Systems Interconnection Conformance Testing Methodology and Framework; the Tree and Tabular Combined Notation (TTCN) (Ed. 2++), May 1999.
4. ETSI ES 201 873-7. Methods for Testing and Specification (MTS); the Testing and Test Control Notation Version 3; Part 7: Using ASN.1 with TTCN-3, April 2012.
5. ISO/IEC 9075. Database Language SQL (Structure Query Language), July 30, 1992.
6. ETSI ES 201 873-1. Methods for Testing and Specification (MTS); the Testing and Test Control Notation Version 3; Part 1: TTCN-3 Core Language, April 2012.
7. ETSI ES 201 873-2. Methods for Testing and Specification (MTS); the Testing and Test Control Notation Version 3; Part 2: Tabular Presentation Format, April 2012.
8. ETSI ES 201 873-3. Methods for Testing and Specification (MTS); the Testing and Test Control Notation Version 3; Part 3: Graphical Presentation Format, April 2012.
9. ETSI ES 201 873-4. Methods for Testing and Specification (MTS); the Testing and Test Control Notation Version 3; Part 1: TTCN-3 Operational Semantics, April 2012.
10. ETSI ES 201 873-5. Methods for Testing and Specification (MTS); the Testing and Test Control Notation Version 3; Part 5: TTCN-3 Runtime Interface (TRI), April 2012.
11. ETSI ES 201 873-6. Methods for Testing and Specification (MTS); the Testing and Test Control Notation Version 3; Part 6: TTCN-3 Control Interface (TCI), April 2012.
12. ETSI ES 201 873-8. Methods for Testing and Specification (MTS); the Testing and Test Control Notation Version 3; Part 8: The IDL to TTCN-3 Mapping, April 2012.

13. ETSI ES 201 873-9. Methods for Testing and Specification (MTS); the Testing and Test Control Notation Version 3; Part 9: Use XML with TTCN-3, April 2012.
14. ETSI ES 201 873-10. Methods for Testing and Specification (MTS); the Testing and Test Control Notation Version 3; Part 10: TTCN-3 Documentation Comment Specification, April 2012.
15. ETSI ES 202 781. Methods for Testing and Specification (MTS); the Testing and Test Control Notation Version 3; TTCN-3 Language Extensions: Configuration and Deployment Support, April 2012.
16. ETSI ES 202 784. Methods for Testing and Specification (MTS); the Testing and Test Control Notation Version 3; TTCN-3 Language Extensions: Advanced Parameterization, April 2012.
17. ETSI ES 202 785. Methods for Testing and Specification (MTS); the Testing and Test Control Notation Version 3; TTCN-3 Language Extensions: Behavior Types, April 2012.
18. I. Schieferdecker and G. Din. A metamodel for TTCN-3. In *1st International Workshop on Integrated Test Methodologies. Colocated with 24th International Conference on Formal Description Techniques (FORTE 2004)*, Toledo, Spain, September 2004.
19. IETF RFC 2616. Hypertext Transfer Protocol (HTTP/1.1), June 1999.
20. ITU-T X.680. Data Networks and Open System Communications. OSI Networking and System Aspects—Abstract Syntax Notation One (ASN.1), July 2002.
21. W3C. Web Services Description Language (WSDL) 1.1, 2001. Available at: http://www.w3.org/TR/wsdl [accessed March 15, 2001].
22. Testing Technologies. TTworkbench—A TTCN-3 tool Set, 2012. Available at: http://www.testingtech.com
23. OMG. UML Testing Profile (UTP) Specification, Version 1.0, July 2005.
24. ETSI ES 202 782. Methods for Testing and Specification (MTS); the Testing and Test Control Notation Version 3; TTCN-3 Language Extensions: TTCN-3 Performance and Real Time Testing, July 2010.
25. ETSI TTCN-3. Home Page, 2012. Available at: http://www.ttcn-3.org
26. I. Schieferdecker and J. Grabowski. The graphical format of TTCN-3 and its relation to UML and MSC. In *3rd International Workshop on SDL and MSC—Telecommunication and Beyond, SAM 2002*, Aberystwyth, UK, June 2002.
27. I. Schieferdecker and T. Vassiliou-Gioles. Realizing distributed TTCN-3 test systems with TCI. In *IFIP 15th International Conference on Testing Communicating Systems—TestCom 2003*, Cannes, France, May 2003.
28. G. J. Myers. *The Art of Software Testing*. John Wiley & Sons, 2004.
29. P. Xiong, R. L. Probert, and B. Stepien. An Efficient Formal Testing Approach for Web Service Testing with TTCN-3, 2005. Available at: http://www.site.oottawa.ca/bernard/softcom paper.pdf
30. I. Schieferdecker, E. Bringmann, and J. Grossmann. Continuous TTCN-3: Testing of embedded control systems. In *Proceedings of the 2006 International Workshop on Software Engineering for Automotive Systems (SEAS '06)*, pp. 29–36. ACM, New York, 2006.
31. H. Neukirchen, B. Zeiss, J. Grabowski, P. Baker, and D. Evans. Quality assurance

for TTCN-3 test specifications. *Software Testing, Verification and Reliability*, 18:71–97, 2008.

32. B. Zeiß. Quality assurance of test specifications for reactive systems. PhD thesis, Universität Göttingen, June 2010.
33. G. Din, D. Vega, and I. Schieferdecker. Automated maintainability of TTCN-3 test suites based on guideline checking. In *6th IFIP Workshop on Software Technologies for Future Embedded and Ubiquitous Systems (SEUS 2008)*, Capri Island, Italy, October 2008.
34. D. Vega, G. Din, and I. Schieferdecker. TTCN-3 test data analyser using constraint programming. In *19th International Conference on Systems Engineering (ICSENG 2008)*, Las Vegas, NV, August 2008.
35. J. Grossmann, D. Serbanescu, and I. Schieferdecker. Testing embedded real time systems with TTCN-3. In *2nd International IEEE Conference on Software Testing, Verification, and Validation (ICST 2009)*, Denver, CO, April 2009.
36. S. C. C. Blom, T. Deiß, N. Ioustinova, A. Kontio, J. C. van de Pol, A. Rennoch, and N. Sidorova. TTCN-3 for distributed testing embedded software. In Perspectives of Systems Informatics, June 27–30, 2006, Novosibirsk, Russia, 2007.
37. OMG. Common Object Request Broker Architecture (CORBA) Specification, Version 3.1, January 2008.
38. Oracle. Java 4 SE Development Kit Version 4, JDK1.4, Documentation, 2005. Available at: http://www.oracle.com/technetwork/java/index.html
39. ETSI ES 203 915. V1.1.1: OSA/Parlay Specification, April 2005.
40. Z. R. Dai. An Approach to Model-Driven Testing—Functional and Real-Time Testing with UML 2.0, U2TP and TTCN-3, Promotion, TU Berlin, Fakultät Elektrotechnik und Informatik, June 2006.
41. 3GPP TS 23.228. Technical Specification Group Services and System Aspects: IP Multimedia Subsystem (IMS), Stage 2, 2006.
42. IETF RFC 3261. Session Initiation Protocol (SIP), June 2002.
43. ETSI TS 102 027-3. Ver. 3.2.1: Methods for Testing and Specification (MTS), Conformance Test Specification for SIP (IETF RFC 3261), Part 3: Abstract Test Suite (ATS) and Partial Protocol Implementation eXtra Information for Testing (PIXIT) proforma, 2005.
44. G. Din. A workload realization methodology for performance testing of telecommunication services. PhD thesis, TU Berlin, Fakultät Elektrotechnik und Informatik, September 2008.
45. GTB TTCN-3 Certificate. The TTCN-3 Syllabus and Certificate, 2008. Available at: http://german-testing-board.info/de/ttcn3_certificate.shtm

第 11 章 主动自动机学习的实际应用

Falk Howar
负责系统编程，多特蒙德工业大学，多特蒙德，德国
Maik Merten
负责系统编程，多特蒙德工业大学，多特蒙德，德国
Bernhard Steffen
负责系统编程，多特蒙德工业大学，多特蒙德，德国
Tiziana Margaria
负责服务和软件工程，波茨坦大学，波茨坦，德国

11.1 简介

目前所用的大多数系统都缺乏全面的规范或充分使用规范，甚至包括未指定的组件。实际上，广为流传的基于组件的软件设计风格通常都是非规范的系统，这是因为大多数的库只能提供组件的很少部分规范。此外，通常的修改和最后一刻的改变都几乎不能列入系统规范。在通信领域可观察到这种困境：通信协议的规范通常是作为与实际实现没有形式化连接的自然语言文档。这就阻碍了任何一种类型的形式化验证方法的应用，如基于模型的测试[14]或模型校验[17]，且难以及时更新文档。实际上，在实践过程中，这些规范根本不时常更新，而使得所有系统很快成为遗产系统。为此，提出自动机学习技术通过允许构建并随后自动更新行为模型来解决这种问题[33-35,44]。这很早之前就在计算机电话集成系统（CTI）[27,28]的具体设置中进行了阐述。在此，可通过观察系统行为对测试队列的反应。在该设置下，行为可理解为由分别表示激励和反应的符号组成的字母表作为语言关键字的测试激励和相应反应组成的队列。自动机学习的目的是为了创建有限状态自动机来描述/近似该语言。

图 11.1 给出了学习过程的一般设计方案，其中将其建模为文献[38]中介绍的扩展过程描述示意图（XPDD）。由图可知，学习过程需要一个用于学习系统（SUL，左侧）的抽象推理字母表和测试驱动程序，并在提交最终模型（右侧）之前产生一系列假设自动机。字母表和一个合适的测试驱动程序是系统需要事先了解的信息，相比于其他常用的基于信息追溯的方法，这种要求是相当低的。在中间部分主要描述学习过程本身：自动机学习通过交替局部搜索（建立封闭性和一致

性）和测试（近似）等效性（如通过一致性测试）产生相应模型。由失败等效测试产生的反例用于引导下一轮的局部搜索。只要激励和反应所选择的解释导致一个确定性语言，则这种技术非常适用于反应式系统。对于这种系统，自动机学习可看作*常规外推法*，即构建一个与观测保持一致的“最佳”常规模型的技术。这类似于众所周知的多项式插值法，其中用多项式代替有限自动机，用函数代替反应系统。正如在此，外推的质量而非适用性取决于所考虑系统的行为结构。然而，由于反应系统中固有的大量自由度，自动机学习的计算成本要比多项式插值法大得多。因此，自动机学习在实践中的成功很大程度上取决于所采用的开发所学系统具体参数的优化方法。一个成功的应用场景请参见文献［42，44］。

图 11.2 给出了实践中如何进行学习设置的一个高层概述：一个学习算法（图中右侧）通过一个测试驱动程序与一个 SUL 实例相连（图中左侧），其中利用系统装置（如由接口描述提供）在学习过程中进行实际系统的调用。测试驱动程序也在学习算法领域中的抽象符号和 SUL 领域中的具体调用之间具有调节作用。在 11.4 节中将进行详细介绍。

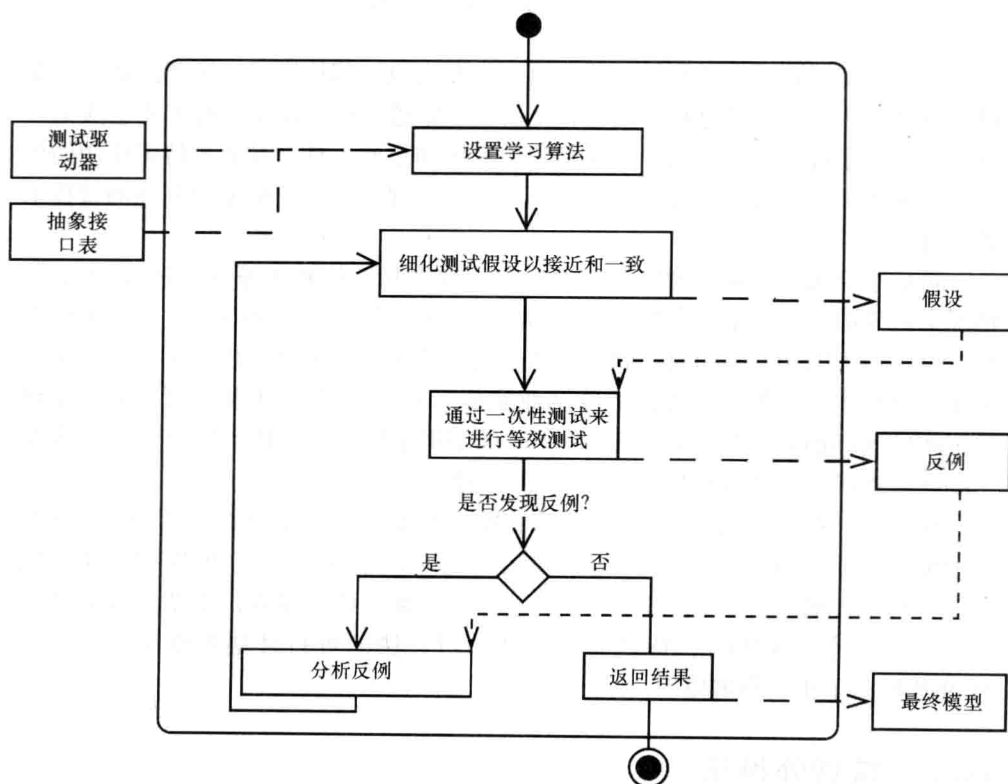


图 11.1　外推算法的常见结构（建模为 XPDD[38]）

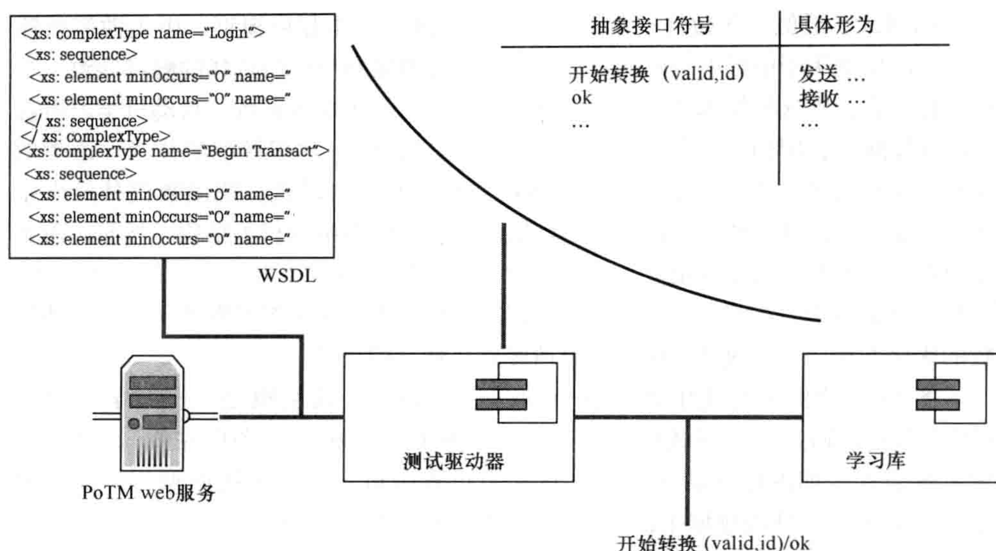


图 11. 2 一个示例设置的示意图

本章总结回顾了自动机学习的本质、主要挑战、其他形式、可能的解决方案以及案例分析说明。并分析得出理论上研究非常透彻的主动学习技术将会成为一种用于实际系统开发的特别的增强型应用程序和重要工具。这需要特定应用的优化方法来提高可扩展性，参数和数据流的充分处理、适合的抽象机制和测试技术的充分适用性。

再如主要关注连接器和协议，即行为与数据相互依赖的系统的连接项目[37]情况下的受限应用场景具有特别的发展前景。在 11.4 节中介绍了一个从类似参数的路由器中成功外推超过 20000 个状态的行为模型的案例分析。在另一个实际的案例分析中，表明如何通过运营企业实施（见第 8 章）外推出到目前为止仍然很小的 Springer 在线会议系统（OCS）的编辑系统行为视图。所有这些都表明主动学习都可能支持这种场景下应急行为的控制。

接下来，在 11.2 节中简要介绍了常规外推原理的基本形式以及 Mealy 状态机模型。在 11.3 节中，简单介绍了实际（反应）系统的学习/外推模型所面临的巨大挑战，随后在 11.4 ~ 11.7 节中进行了详细讨论。接着，本章继续介绍了下一代学习库（NGLL），特定应用领域学习算法的可行性建模框架，然后在 11.9 节给出了小结和展望。

11.2 常规外推法

常规外推法试图构建一个确定性有限状态自动机（DFA），该自动机与一个

目标给定自动机的行为在给定自动机观测及其内部结构的某些额外信息基础上相匹配。在此，只是总结了实现过程中的一些基本方面，这是基于文献［4］中的 Augluin 的学习算法 L^*。

定义 1　一个 DFA 是一个元组

$$A = \langle Q, q_0, \Sigma, \delta, F \rangle$$

式中，Q 为一个有限状态的非空集合；$q_0 \in Q$ 为初始状态；Σ 是一个有限字母表；δ：$Q \times \Sigma \rightarrow Q$ 是转换函数；$F \subseteq Q$ 为接受状态的集合。

直观上，DFA 是通过状态 $q_0 \in Q$ 演化而来的。只要利用了一个输入符号（或行为）$a \in \Sigma$，自动机就会根据 $\delta(q, a)$ 转换到一个新的状态。当且仅当 DFA 在从其初始状态开始处理 w 后达到一个接受状态 $q_i \in F$，则 DFA 就会接受字符 $w \in \Sigma^*$。记 $q \xrightarrow{a} q'$ 来表示在输入信号 a 上，DFA 从状态 q 转换到状态 q'。

L^* 算法也称为一种主动学习算法，可以通过主动查询 SUL 来学习 DFA。该算法提出*归属查询*来测试特定字符串（潜在运行）是否包含在 SUL 语言（运行集合）中，以及*等价查询*来比较与 SUL 等效的语言中构建的中间假设自动机。一旦成功产生一个等价查询信号，则学习过程成功结束。

从实用的角度来看，这些方法可看作一种基于目标系统简单接口的算法模式来提出隶属度查询和等价查询。通常，隶属度查询可通过测试来实现，而等价查询必须在隶属度查询的基础上不断逼近。这意味着所要学习的目标自动机在这一原则下准确定义：这是由查询结果定义的最小的确定性自动机。特别声明的是，这并不保证是正确的或完备的。相反，其精度很大程度上取决于等价查询的近似程度。在等价查询完美且目标系统规则的情况下，自动机学习可保证以目标系统语言规范的最小确定性自动机终止。

在其基本形式上，L^* 从一个用于处理与考虑字母表（基本观测）类似的所有字符的假设自动机开始。也就是说，其具有一个单个状态并在查询结果的基础上完善该自动机，然后迭代执行如图 11.1 所示的两个主要步骤：①通过隶属度查询建立封闭性和局部一致性；②通过等价查询测试等价性。由此可见，该过程会连续产生最小状态的确定性（假设）自动机，且与所有查询结果保持一致。能够获得该结果的关键在于基于著名的 Nerode 一致性的 L^* 的状态双重表征[36]，这可通过所谓的残差语言[20]来确定最小语言受体的状态，即表征相对于可能前缀的接受的可证明的有限语言集合：

- from below，是一组*访问序列*（或可能的前缀）S。这些状态的特点是可能过于完美，因为不同的字符 s_1，$s_2 \in S$ 可很好地 Nerode 同余。也就是说，可以接受相同语言作为可能的后缀。L^* 会创建这样一个包含对目标自动机所有状态的访问序列的集合 S。另外还包含第 2 个集合 $S \cdot \Sigma$，该集合和 S 集合可覆盖 SUL

的所有转换。

• frome above，这是通过对残差语言的有限预测，也就是说，对作为对所考虑语言字符给定一个前缀下完成后缀集合的语言的有限预测。这个特点可能过于粗糙，因为有限预测可识别两个截然不同的残余语言。然而，可表明通过至多 $n-1$ 个可能后缀能够区别 n 种不同的残余语言。这样一种（尽管一般并非最小）有限预测的相应结构可认为是 L^* 的核心。该结构的作用是动态构建一个增长的（后缀）字符集合 $D=<d_1, \cdots, d_k>$ 作为预测的目标，其中 $d_i \in \sum^*$。L^* 可实现 from above 的状态 q 的特征，然后简化为位矢量 $row(s)=<r_1, \cdots, r_k>$，其中 $r_i \in \{T, \perp\}$，用于表示相应的字符 $d_i \in D$ 是否属于 q 的残差语言：

$$r_i \in row(s)=T \Leftrightarrow <d_i \in L_q \Leftrightarrow s \cdot d_i \in L>$$

式中，L 表示 SUL 的语言；L_q 表示描述 q 的残差语言；s 假设为从初始状态达到 q 时来自于 S 的一个字符。

第 2 个重要特点是构建（中间）假设自动机的关键：每个发生的位矢量都对应于其中的一个状态。这需要非常小心以确保这种结构可产生定义良好的确定性自动机。

最初的假设自动机只含有一个状态且由空观察 ε 的隶属度查询输出描述，S 和 D 都初始化为 $\{\varepsilon\}$。从而在空字符在语言中的情况下，可接受任何字符，否则无字符。图 11.1 所示的学习过程是按下列执行的：

局部探索。第 1 步是两个阶段的再次迭代。第 1 阶段是检查构建的自动机是否在一步转换下是闭合的，即从假设自动机的每个状态开始的每次转换都终止于该自动机中一个明确定义的状态。即对于每个 $r \in S \cdot \sum$，存在一个 $row(s)=row(r)$ 的 $s \in S$。否则，S 集合将由相应的 r 进行扩展，直到实现封闭性。该扩展是保证产生一个唯一的固定点，在所处理的行中顺序无关。

第 2 阶段是检查具有 from above 特点 $row(s_1)=row(s_2)$ 的相同位矢量的两个访问序列 s_1，$s_2 \in S$ 是否具有相同的输出转换，同时一个必须的前置条件来表示相同状态。这个称为一致性的条件可形式化表示如下：

$$\forall s_1, s_2 \in S \, \forall a \in \sum \cdot row(s_1)=row(s_2) \overset{?}{\Rightarrow} row(s_1 \cdot a)=row(s_2 \cdot a)$$

由上，可很容易地看到，可通过一种使得与转换前具有明显差别的可区别转换变量的形式来描述不同特征集 D，以此来去除检测到一致性：只需对可通过该转换标签来分离可区别转换上的两个目标状态的区别特征增加前缀。

但遗憾的是，这种对 D 的增加可能会破坏之前的完整性，因此需要重新迭代完成过程。从另一方面来说，这可能又会违背一致性。然而，这两个步骤的连续迭代可保证产生一个独有的、定义良好的、封闭的完备假设自动机，其状态均由位矢量表征。具体如下：

• 假设自动机的每个状态 $q \in Q$ 都至少在一个字符 $s \in S$ 下可达，即 $row(s)$ 对应于 q。

• 如果通过一个 L 的字符可达到，则认为 $q \in Q$。

• 当且仅当存在 $s \in S$（*证据*）且 s 达到 q（或 $row(s)$ 与 q 相对应）和 $s \cdot a$ 达到 q'，则存在一个转换 $\delta(q,a) = q'$。

等价检测。当建立封闭性和一致性之后，等价查询可检测假设自动机的语言是否与 SUL 的语言一致。如果成立，则学习过程成功终止。否则，等价查询返回一个反例结果，即可区别假设和 SUL 的一个字符。

反例处理过程提供了一种新的访问队列和新的不同前缀来完善假设自动机（见图 11.1）：在基本版本中，在开始新一轮建立封闭性和一致性之前，该反例的所有前缀都增加到 S 中。这是为了保证至少存在一个会对假设自动机引入至少一个附加状态的不一致性。

该方法的正确性验证可按照以下一个不仅适用于 L^*，而且适用于所有已知衍生算法的简单模式[39,41,42,50,51]：

1）通过残余语言预测的状态结构可保证假设自动机中的状态个数不会超过满足所考虑语言的最小确定性自动机的状态个数。

2）闭包过程可保证假设自动机的每次迁移都具有一个访问队列。这意味着假设自动机的规模（状态个数）和所考虑语言的最小确定性自动机必须同构。

3）反例的处理可保证对于每个反例，至少对假设自动机会增加一个附加状态。但由于 1）和 2）的原因，这种处理一般只能有限发生。

4）一旦假设自动机的语言与期望结果不一致，等价检测机制（也称为等价数据库）会提供一个新的反例。

综上，这样就可以保证在经过至多 n 次等价查询且在所考虑语言最小确定性状态机的情况下（其中 n 为该状态机的状态个数），学习过程终止。

11.2.1　充分行为建模

主动自动机学习最初已引入到有限状态的受体中[4]。在此阐述在通信系统相关文献和验证中自动机学习相关的实践[27,28]。然而，与有限状态受体不同，反应系统不能区别接受状态和拒绝状态，但会产生一些相对于输入的输出响应。为满足这一要求，将自动机学习转换为 Mealy 有限状态机[42]。

定义 2　一个 Mealy 有限状态机定义为一个元组

$$S = \langle Q, q_0, \Sigma, \Omega, \delta, \lambda \rangle$$

式中，Q 为有限非空状态集合（其中 $n = |Q|$ 为 S 的大小）；$q_0 \in Q$ 为初始状态；Σ 为有限输入字母表；Ω 为有限输出字母表；δ：$Q \times \Sigma \to Q$ 迁移函数；λ：$Q \times \Sigma \to \Omega$ 输出函数。

直观上，Mealy 有限状态机可通过状态 $q \in Q$ 不断演化，只要应用一个输入符号（或行为）$a \in E$，则状态机可根据 $\delta(q, a)$ 转移到一个新的状态，并根据 $\lambda(q, a)$ 产生一个输出。记 $q \xrightarrow{i/o} q'$ 表示在输入符号 i 下 Mealy 有限状态机从状态 q 转移到状态 q'，产生输出符号 o。

转移函数 δ: $Q \times \Sigma \rightarrow Q$ 可扩展为 δ': $Q \times \Sigma^* \rightarrow Q$，使之适用于所有状态 q，$q' \in \Sigma$。字母 $a \in \Sigma$ 和字符 $w \in \Sigma^*$，且满足 $\delta'(q, aw) = \delta'(\delta(q, a), w)$。同样，也适用于输出函数。在本章后面，通过 a 和 λ 根据上下文来表示这些函数的扩展形式或原始版本。

Mealy 有限状态机已广泛应用于确定性反应式系统的建模中，且仍在不断开发新的 Mealy 有限状态机学习算法[49,51]。实际上，Mealy 有限状态机学习在实践与大规模应用中占主导地位。现在，已有一些网络服务[47]、通信协议实体[11]或软件组件[48,52]的行为模型学习示例。在 11.4 节，将介绍一个软件组件学习的示例。从概念上来讲，Mealy 有限状态机的学习是按照上节所述的相同模式。唯一的不同之处在于区别接受或拒绝字符的双态鉴别器由一组鉴别转移的输出符号来代替。这使得在所需数据结构方面几乎不存在技术上的修改，同时不会影响之前的整体正确性、完备性和终止参数。

最近，推理方法的发展主要集中进一步收集在实际系统中可能发生的现象。在 Mealy 有限状态机的推理算法基础上，产生了 I/O 自动机推算法[3]、时间自动机[26]、Petri 网[21]和消息序列图[12,13]。在 I/O 自动机模型基础上，大量由静止状态组成的系统可适用于查询学习。时间自动机对时间相关行为进行显式建模。利用 Petri 网可解决具有显式并行状态的系统。在此，将在一节中专门介绍这种参数化接口字母表的复杂问题的解决方法（11.7 节）。在文献[9]中，将主动学习应用于具有复杂行为且参数在由无限状态空间组成的无限域中的系统。

除了对学习算法进行扩展来涵盖更广泛的现象之外，在模型校验中应用主动学习也是一个研究热点领域[18,40,46]，尤其是克服所谓的空间状态爆炸问题。

11.3 常规外推法的挑战

自动机学习可认为是处理*黑箱系统*的一种关键技术，即可进行观察，但对其内部结构甚至系统目的相关知识完全未知或已知很少的系统。主动自动机学习的特点在于观察的特殊方法，即以一种进行隶属度查询和等价查询的积极主动方式。为此，需要某种方法来实现应用背景下基于查询的交互。鉴于隶属度查询通常是通过测试在实践中实现的，而等价查询通常是不切实际的。在接下来的内容中将会讨论由于应用场景的各种不同特点所面临的挑战，并阐述了实际黑箱场景

中“黑并不等于黑”的问题。

A：与实际系统交互

与一个实际 SUL 系统交互会产生两个问题：一个是建立合适接口使之应用于实现隶属度查询的测试案例时的纯粹技术问题；另一个是缩小抽象学习模型和具体运行场景之间差距的概念性问题。

第 1 个问题对于系统连通性设计（如网络服务或代码库）而言相对简单，该问题具有一个由外部触发且如何处理所产生文档的基本概念。但是，如对于在封装良好的环境下工作的某些嵌入式系统，连通性的建立可能会非常复杂，且只能通过一些专门的图形用户接口（GUI）来访问。

第 2 个问题在概念上更具有挑战性。该问题涉及根据通信字母表建立一个合适的抽象层，这一方面可产生一个有用的模型结构，另一方面也允许在抽象模型层和具体 SUL 层之间自动地来回转移。但找到一个合适的抽象并不容易。例如，一个比较大的问题是可能易于产生非确定性问题，由此影响学习过程。近年来，这个问题已通过一种自动字母表抽象细化方法得以解决[31]。

B：隶属度查询

尽管某些规模较小的学习实验一般只需要几百个隶属度查询，但实际学习系统可能需要比其多好几个数量级的隶属度查询。这直接表明在进行隶属度查询时或在相应测试案例中大多数实际设置时 SUL 的执行速度是极其重要的。对于这种类型的系统，仿真环境下通常每秒可进行数千次查询，而在实际系统中每个测试案例可能需要很多秒或甚至几分钟。在这种情况下，所需测试案例个数最小化，而非并行化，是成功与否的关键（见 11.5 节）。

C：重置

主动学习要求隶属度查询是独立无关的。尽管这在仿真系统中完全没有问题，但在实际系统中可能存在很大的问题。在此，解决方案可以从通过系统状态的自动归位序列[50]或映射的重置机制到产生独立的全新系统场景之间变化。事实上，在某些情况下，可执行的一种最佳办法是对于一个完全独立的用户场景下进行每次隶属度查询（见第 8 章）。除了建立这些场景的成本之外，还要求对查询结果进行充分地聚类。例如，各种场景下不同的用户密码组合必须经过抽象的确定。

D：参数域和值域

主动学习一般是基于抽象的通信字母。参数和解释值只能处理为抽象字母中一定程度的表示。在实践中，这通常对于甚至是简单通信协议的系统都是不充分的，例如必须控制增加的序列个数，或需要验证用户名/密码是否匹配。由于该问题的复杂性，并不期望在此得到任何全面的解决方案。而认为需要开发特定域和特定问题的解决办法来产生专用的解决方案。

以下内容中将致力于解决这些挑战。对于每个挑战，将会提供一个相应结果的简单描述和一个小的实际示例。

11.3.1 等价查询注释

所面临的一个巨大挑战是如何实际实现等价查询，在本节中并不进行详细讨论。这个方向的相关研究很少，在此并不利用一节来专门讨论该问题，而是将目前为止所相关文献中进行的尝试进行总结。

等价查询是将学习的假设模型与语言等效的 SUL 进行比较，并在失败的情况下返回一个表示差异的反例。在仿真环境下这种实现相当容易：如果 SUL 是一个模型，则可显式检测是否等价。然而在实践中，SUL 通常是某种类型的黑箱，必须利用隶属度查询来模仿等价查询。一般来说，这样只会产生一个近似解决方案，因为总是会存在没有充分检测的可能性。

基于模型的测试方法[14]已用于模拟等价查询。例如，如果已知 SUL 中状态个数的上界，则可采用 W 方法[16]和 Wp 方法[22]。这两种方法都具有指数复杂度(在 SUL 的大小和所需的隶属度查询个数方面)。在文献[6]中曾最先讨论过常规外推法和一致性测试方法之间的关系。

在没有引入任何额外假设的情况下，只存在进行隶属度查询的近似解决方案。在此，一致性测试方法可能并不总是一种明智选择。这就需要从如通过采用一致性测试技术的“试图证明等价”转变为具有积极影响的“快速发现反例”。近年来，Zulu 的挑战[19]在该方向上进行了深入研究，并在文献［30］中进行了阐述。

11.4 与实际系统交互

在本节中，将集中讨论系统交互的第 2 个问题：缩小抽象模型层和具体（实际）SUL 层之间的差距。

一种主动学习算法可利用其通信字母表产生抽象测试序列，该通信字母表需转换为 SUL 的具体测试序列。在响应上，学习算法期望得到一个在通信字母表层次上的抽象输出序列，这通常是一个相对于具体测试序列的输出产生抽象。实现这种来回迁移的组件在不同背景下有不同的名字，例如映射器、测试驱动程序或传感器。为强调对于一致性测试设置的近似，在此将其称为测试驱动程序。

尽管实际系统的具体测试仪器之间存在差异，但可以观察到如下的一般模式。一个学习算法可利用一组 SUL 的（抽象）输入符号进行实例化。学习过程中算法程序的隶属度查询（输入符号序列）可传送到测试驱动程序中。然后，测试驱动程序通过以下步骤来对系统进行操作：

- 将隶属度查询转换到实际系统的行为序列中；

- 在 SUL 中逐一执行这些行为；
- 记录每个行为所产生的反应；
- 将这些反应转换为学习算法可理解的输出字母表符号。

在输入序列执行完成后，输出的结果序列将会传送回学习算法。在作为一个学习设置的一般拓扑结构示例中已讨论过的图 11.2 中描述了一种推理网络服务行为的相应设置。在这种情况下，在服务的网络服务描述语言（WSDL）文件中可获得字母表信息，且测试驱动程序在对应于算法所用抽象接口符号的具体行为下工作。

测试驱动程序是处理抽象（见 11.7 节）或在学习算法[2,3]中 SUL 完全隐藏部分的一种功能强大的方法。然而，设计这样一种测试驱动程序并不是一个简单工作。由于将会对由学习算法产生的实际系统产生显著影响，因此需要专业知识。到目前为止，大多数测试驱动程序都是人工定制的。据现有知识所知，仅在文献[37]中明确提出了主动学习测试驱动程序的自动生成。

11.4.1 测试驱动程序设计示例

接下来，简单介绍一个应用自动机学习的实际系统。在文献[48]中给出该案例分析的详细讨论。这是一个最初由 Verona 大学在系统 C 中设计的用于硬件/软件协同设计的简单路由器。该路由器提供了 4 个可用来接收和发送数据包的端口。其基本组成一个用于缓存消息的有界 FIFO 序列和一个包含如何路由消息的信息表。消息本身包括发送地址、接收地址、发送方生成的识别码、数据和校验码。

在设计测试驱动程序时，将仅将识别码、数据和校验码上不同的消息通过采用一个静态路由表（其中，路由目标 0 到端口 0，路由目标 1 到端口 1，以此类推）来看作等效数据包。只要路由器在某一端口接收到一条消息，则将该消息压入队列尾部，直到队列满。若消息队列已满的情况下接收到一个数据包，则该消息丢弃。

在此，也清晰地体现了充分建模的复杂性。将发送给路由器的消息看作输入字母表而返回的响应消息看作输出字母表的清晰定义不成立，除非至少关注缓冲区的边界性能。这种建模方式会使得对于每个不同缓冲区都建立相同模型，这是因为学习算法的建模会对路由器发送一个输入数据包，然后在发送新的数据包之前等待，直到学习算法从路由器中接收到一条消息。

该问题可通过分离输入行为和输出行为使得发送操作和接收操作不同步来解决。更具体而言，就是将输入分为接收和路由两个行为后，只要激活接收行为，则路由器在某一端口处接收消息，并尝试将其保存在数据包队列中。该输入行为总是由一个具有使得学习机发送额外数据包的 void 输出的测试驱动程序响应，而与路由器是否可以排序数据包无关。另一方面，路由行为表示了从队列中取消息并将其路由到某一网络端口。如果队列为空或数据包发送到某一指定发送输出端口时，则输出响应为 nop。

由于该路由器提供了4个端口，因此可利用16个涵盖每种源地址和目的地址组合的不同消息数据包进行分析。在此，对不同类型的路由器采用自动机学习方法，其中消息队列的长度可在1~7变化。图11.3给出了由学习算法创建的消息队列长度为1的系统最小模型的图形化表示。这个迁移过程由观测的输入/输出标记。

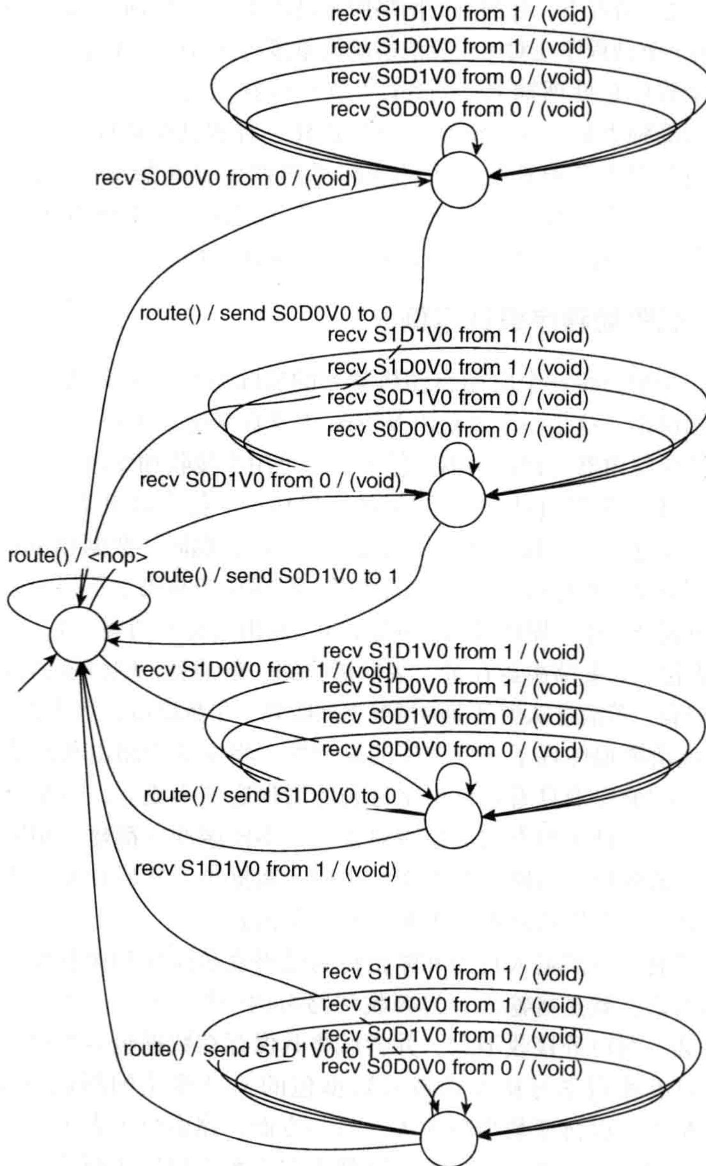


图11.3　队列长度为1~4个端口的路由器模型

可明显看出，路由器模型的消息队列长度以指数形式增大。队列长度为 7 的最大学习模型具有 21844 个状态。据我们所知，这是所学习的最大“实际”模型[48]。目前为止，根据 LearnLib 学习的最大系统是一个具有从开放/Caesar 工具套件的 3 个电话的普通旧电话系统模型[23]。该模型具有 39974 个状态、51 个输入和大约 200 万个迁移。

11.5　隶属度查询

学习算法广泛应用隶属度查询来系统地探索一个系统行为，直到不再检测到更新模型和系统行为之间一致性的“直接证据”。

为减少探索系统所必须的队列个数，提出了一些原始 L^* 算法的扩展算法[4]。

这些扩展方法包括从去除局部一致性检查[41]、优化最坏情况下查询个数[50]到全部替换数据结构[39]或所用自动机模型[3,35]。在文献[5]中给出了部分概述。

另一种减少在系统上需执行测试个数的方法是采用特定应用程序的滤波器。这些滤波器可显著减少隶属度查询的个数并开放对实际系统的常规推理。接下来，将简要讨论一些提高形式化的有效但对于滤波而言一般思想的方法。相应的实验是基于一个 DFA 的设置，但结果可直接应用于第 8 章所述的 Mealy 有限状态机的更一般设置。

11.5.1　冗余度

在系统地探索系统行为能力的经典实现过程中，学习算法可产生冗余的隶属度查询，即对于相同的测试案例产生不同的推导结果。为防止自动测试装置执行两次测试案例，可利用缓存来检测是否执行了两次并将其滤除[42]。

另外，在类似测试的场景中，隶属度查询将会返回所有轨迹，而不仅仅是单个符号（如有限状态受体下接受或拒绝的情况）。因此，在实际中进行隶属度查询通常可产生仅在字符所有前缀条件下进行隶属度查询所获得的理论上形式化的信息。这些额外信息也可通过缓存方式进行保存。

11.5.2　前缀闭包

如果所学习的语言由对实际系统运行的观测组成，则很显然，该语言是前缀闭包的，即给定一次运行，该次运行的每个前缀也在该语言中（因为，前缀本身也是系统的一次运行）。由于学习算法无需考虑通过之前的隶属度查询方法从 SUL 语言中排除的字符串是否连续，因此这个观察结果可以产生一种强大的优化。另外，只要在系统运行中已知一个长字符串，就可将该字符串的所有前缀都添加到模型中而无需进一步测试。按形式化的正确性术语，如果一组未接受状态

Q、F 不能舍弃，则 DFA 为前缀闭包，即

$$\forall q \in (Q \backslash F) \cdot \forall a \in \Sigma \cdot \delta(q,a) \in (Q \backslash F)$$

图 11.4a 给出了一个前缀闭包的 DFA：底部状态是唯一拒绝的状态，即一个汇节点。值得注意的是，最小化前缀闭包 DFA 的特点是因其仅有一个拒绝状态而具有一个汇节点。

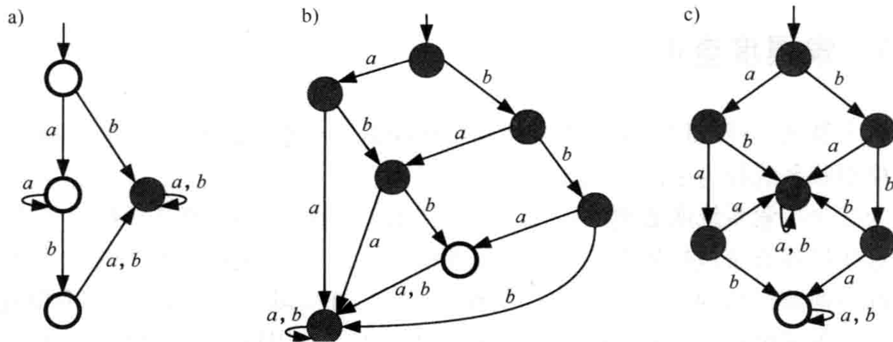


图 11.4　具有不同特点的确定性有限状态机（接受状态具有白色和粗实线）
a）前缀闭包的 DFA　b）相互独立的行为 a，b　c）对称行为 a，b

前缀闭包的理论概念可很容易地应用于 Mealy 有限状态机和实际系统。这种概念有助于忽略 SUL 中的无关部分：例如，在测试驱动程序（见 11.4 节）中仅需要对特殊错误符号屏蔽特定输出，并保证错误输出会将 SUL 转移到一个任意定义的错误汇节点。在文献[7, 35, 42]中给出将一个前缀闭包滤波器应用于实际模型中所产生的结果。

11.5.3　行为独立性

可观察的事件在某种意义上可能是独立的，即以任何顺序执行都产生相同的系统状态。由此，如果观察（或查询）到一个执行顺序，则可得出独立事件的每个重新排序都可产生相同的系统状态。尤其是，如果这些执行顺序中有一个事件是一次系统运行，则独立事件 a，$b \in \Sigma$ 的所有（等价）重新排序也是系统运行：

$$\forall q \in Q \cdot \delta(\delta(q, a), b) = \delta(\delta(q, b), a)$$

独立滤波器可通过只查询系统中每个等价类的一个元素即可得到上述观察。

尽管总是可采用一个前缀闭包滤波器，但一个独立的滤波器需要一些形式化表示用于指定哪些事件可被打乱的独立关系的额外输入。例如，图 11.4b 中的确定性有限状态自动机包含了一对黑色表示的独立行为（a，b）。从形式上讲，独立性是指一对行为之间的一种非自反和对称关系。一个独立滤波器的实现过程可根据独立性关系来规范化查询：即根据行为的给定顺序来计算按字典顺序的最小

等价查询[35,42]。

11.5.4 确定性输入

对于许多实际系统，可根据以下情况来区分输入字母表和输出字母表：总是输入使能，反映了系统无法控制其环境和输入的确定性响应。因此，只要已知一个特定输入序列的响应，则无需进一步考虑该序列。已证明这种优化在将该系统建模为有限状态机自动机时非常有效[35]（见图 11.4）。然而，这也表明通过切换到 Mealy 有限状态机模型可非常自然地实现这种优化。

11.5.5 对称性

硬件和通信系统中通常包含大量的无法通过观察来区别的组件，也就是说，不能显式观察到其识别码。例如，从一个观察的角度来看，一般并不关心何种设备执行某一特定行为（例如，寻址哪个内存组或哪部电话拨打特定号码），并且从原则上相应组件的精确识别（请求处理器或呼叫接收器）并不重要，只要假设一个唯一且一致的识别码，例如呼叫号码和接收器的号码相匹配[42]。

随着系统中相同组件的个数增大，该观察对于优化具有很大的潜力。在此，实现了一个相应的滤波器，在其本质上导致了对设备的一种符号处理。图 11.4c 给出了一个 DFA 示例，其中，认为 a 和 b 对称，即可完全互换。

11.5.6 滤波器示例

现已对文献[42]中呼叫中心应用的几个典型示例进行了模型构建实现。为阐述实验目的，在此给出 4 种简单场景，其中每个场景都包括一个连接到电话号码的电话交换机。在这些场景中，学习模型的重点限于只针对电话的少数行为和交换机的某些响应。在最简单的场景（S1）中，仅允许一个电话接听和挂断，在最后一个场景（S4）中，具有 3 个电话，其中两个电话之间可连接。在文献[44]中给出了设置的具体细节。

由图 11.5 中的示意图可知，系统确定性输入、语言的前缀闭包和特定独立性行为的组合可显著减少测试数量。根据图中给定图例的顺序应用滤波器，并根据涵盖测试案例的第一个滤波器来对测试案例进行分类。值得注意的是滤波器的应用顺序可能会改变组合的效果（见文献[43]）。

前缀滤波器对所有考虑的场景的作用相似。这表明并不依赖于示例性质和状态个数。而其他两个滤波器（确定性输入和偏序）的效果则变化很大：节能因子随着状态个数增大而增大。另外，还期望随着独立性程度提高，偏序和对称性的影响也会增大。这可由实验验证：S1 中仅有一个执行体，因此无独立性可言，使得因子为 1。随着独立设备的个数增加，节能因子也会相应增大，如 S2 和 S3 所

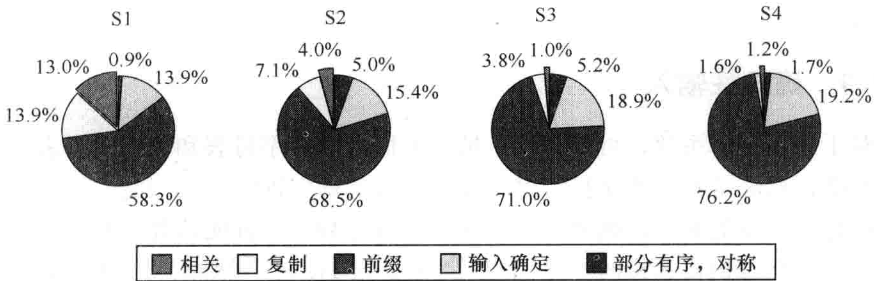


图 11.5 每个滤波器的成员队列百分比滤波

示。与 S3 相比，S4 的节能因子减小。这是因为在 S4 中引入的行为（呼叫初始化）建立了两个设备之间的相互依赖关系，由此减少了偏序和对称性的潜在优化。

11.6 重置

在原则上，学习反应式系统需要一种将 SUL 重置为初始状态的方法：为保证结果有意义，应在相同初始条件下执行所有隶属度查询（或实验）。这种要求相对苛刻，在实际系统中可能并不满足。实际上，可将系统需重置到相同状态的要求简化为将系统重置到一个可观察的等价状态即可。目前已有几种其他方法比显式重置机制更适用于在实际场景中明确重置一个系统：

- 实现重置的一种方法是一个全局有效的自动归位序列，即一个总能使得 SUL 返回其初始状态的序列。对于大多数具有复杂控制结构的软件系统，显然并不在一个全局有效的归位序列。然而，在硬件系统中，这是很常见的。在文献[1]中将一个全局有效的归位序列作为重置机制。分布式软件系统很大程度上取决于消息传递范式，且一个单独的组件并不能构成一个复杂的控制结构。对于这个系统，接收通道中所有消息的行为序列会在一个特定状态下离开该系统。执行该行为序列两次仍会产生相同的“空”状态。

在开始实际的学习过程之前，归位序列通常是人工设计的。实时产生归位序列的推理系统的可能性已成为理论研究的一个热点问题[50]。

- 如果能够控制离线实验的实现过程，例如，存在以某种测试形式运行的 SUL 情况下，通常可以对每次实验简单利用 SUL 的一个新的实例。在文献[2，11，49]中，就利用该方法来推断协议实体的行为。协议实体一般运行在网络模拟器中，对于每次实验都需要创建一个新的协议实体。
- 如果只存在系统的一个实例，可能会利用抽象而不是重置：例如，可利用一个抽象接口字母表来学习每次新连接或用户采用相同行为的系统，其中，在测试驱动程序中，抽象接口字母表转换为测试该性能的具体实验。在文献[1]中介

绍了一个案例分析，在不通过任何方式重置服务的情况下，外推一个网络服务的行为模型。所提方法取决于假设这些服务通常为每一个新的连接提供相同行为。从所用学习算法的字母表中抽象出具体事务标识符，并在每次实验中利用一组新的事务标识符。由此可将具体服务应用到在抽象层不可区别的状态中。该技术用于学习第 8 章所述的 OCS 的全局行为学习。

这种简化显式重置机制要求的简短而不完整的思想表明可以通过多种方法来避免从字面上重置 SUL。

11.6.1　重置示例

在此，给出了一个体现第 3 种实现重置机制方法的案例分析的部分节选。在文献[1]中详细讨论了该案例分析。该案例分析的基本内容是“移动支付”(PoTM) 服务，可为消费者和网店经营者之间提供支付管理功能。只要提供支付的过程细节和账户凭据，PoTM 的用户即可开始交易。另外，可能还会要求一个交易 ID。这种服务提供了两个额外的原语：一个是检验交易状态，另一个是让银行确认交易（以使得交易取消可能）。在图 11.6 中给出了定义网络服务接口的部分 WSDL 文档。

可通过抽象来实现系统重置：个体应用实例之间的差异（其中，每个实例都有各自“新”的交易 ID）变得不可观测。在此，新的交易 ID 是指目前为止在系统中尚未使用的交易 ID。利用返回已使用和未使用交易 ID 之间差异的 check-TransactionStatus 原语来进行是否为新的检测，且不改变系统状态。

```
...
<xs:complexType name="BeginTransaction">
 <xs:sequence>
  <xs:element minOccurs="0" name="transactionId" type="xs:string"/>
  ...
 </xs:sequence>
</xs:complexType>

<xs:complexType name="CheckTransactionStatus">
 <xs:sequence>
  <xs:element minOccurs="0" name="transactionId" type="xs:string"/>
 </xs:sequence>
</xs:complexType>

<xs:complexType name="AknowledgeTransaction">
 <xs:sequence>
  <xs:element minOccurs="0" name="transactionId" type="xs:string"/>
  ...
 </xs:sequence>
</xs:complexType>
...
```

图 11.6　PoTM 网络服务的 WSDL（摘录）

图 11.7 给出了学习算法建立的仅利用新交易 ID 进行支付服务模型的图形化表示。根据相应的输入符号和输出符号来标记状态迁移。所给出的输出符号是激活原语的返回值。返回值“1002”表示成功返回。在存在错误或异常的情况下，网络服务给定的原因表示为输出符号。

由图 11.7 可知，只有利用未使用交易 ID 的一个新交易才会产生一个系统的新状态。所学习的模型证明了允许降低重置要求的假设：对于所有并发的交易，服务行为是独立相同的。

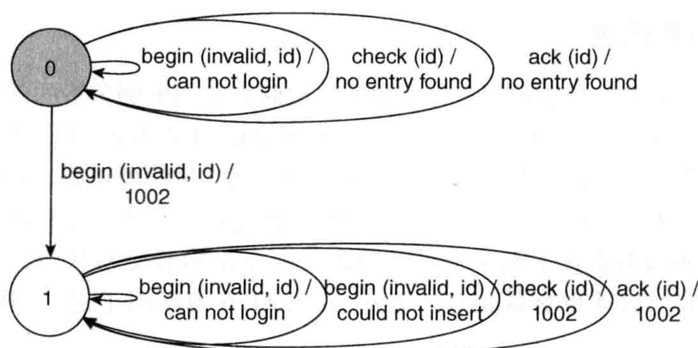


图 11.7 PoTM 网络服务的外推模型

11.7 参数和值域

正如 11.4 节中所述，实际系统的学习模型需要一个测试驱动程序，可用于连接学习算法和 SUL。如上述讨论，测试驱动程序也可用于抽象，例如通过隐藏特定参数。然而在本节中，将考虑自动组成带参数行为（如呼叫方法）的接口字母表的处理方法。对于这些一般系统，不可能推理每个字母符号下具有具体数据值的所有可能实例行为。但是，为了确定这些系统的子类，可采用常规外推法。例如，如果参数影响仅通过布尔表达式在参数上建模的 SUL 行为，这将变得可能。

定义 3 一个参数化的 Mealy 有限状态机定义为一个元组

$$S_P = <Q, q_0, A, \Omega, \Gamma>$$

式中，Q 为一个有限非空状态集（$n = |Q|$ 为 S_P 的大小）；

$q_0 \in Q$ 为初始状态；

A 为参数化行为的有限集；

Ω 为有限输出字母表；

Γ 为迁移有限集合，每次迁移为一个元组 $<q, a(\bar{z}), g, q'>$，其中 q，

$q' \in Q$，$(a(\bar{z}), g)$ 为守护行为。

与 Mealy 有限状态机类似，通过状态吸收输入并产生输出符号可演变出一种参数化 Mealy 有限状态机。$a(\bar{z}) \in A$ 表示一个行为，$\bar{z}$ 为 a 的一个抽象参数矢量。用 $\bar{c}$ 来表示 $\bar{z}$ 的具体值。$a(\bar{z})$ 的守护 g 是一个在单个组件 $\bar{z}$ 上不可分布尔命题的组合。守护行为包括（$a(\bar{z})$，g）行为和守护条件；这是 $\bar{c} \mid = g$ 下所有具体 $a(\bar{z})$ 的抽象表示。在此要求 Γ 是一个定义良好的函数。

参数化 Mealy 有限状态机的学习算法通常初始化为只具有一些具体输入符号和最一般的守护布尔值为真。由此，字母表可动态扩展，同时守护条件也会相应地更加精确。守护条件细化的基础是采用组合推理方法：主动学习成为推理状态空间的一种框架，该状态空间应与策略相结合来推理一个正确的抽象接口字母表，即本例情况下的守护公式构建。在文献[32]中给出这种字母表抽象自动完善技术的一般模式：字母表扩展序的基础是在等价查询中获得的反例，这也表明：

- 当前的假设至少缺少一个状态；
- 其中的一个假设守护过于一般。

当一个反例表明假设守护/字母表符号过于一般时，就会对字母表进行完善。这种情况下，过于一般的守护/字母表符号可分为两部分，并继续以改善后的字母表进行学习。这将会扩展图 11.8 中左侧突出显示的经典学习模式。

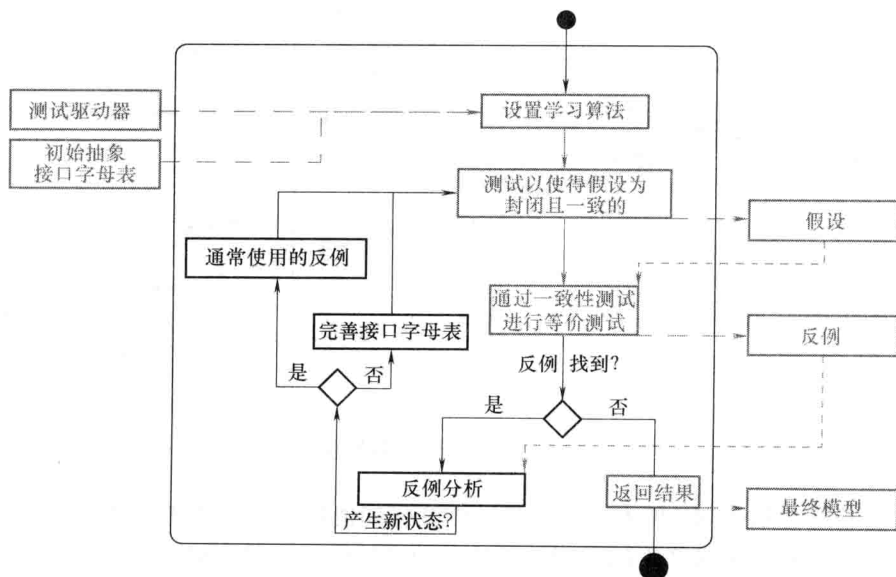


图 11.8　组合外推算法的结构（在 XPDD 中建模[38]）

在文献[8，24]中给出了两种该模式的具体实现：在文献[8]中，提出了一

种通过输入行为和布尔参数推理参数化系统行为模型的方法。在这种情况下，守护条件是在这些参数上的析取范式（DNF）公式：对于一个守护条件涵盖的符号，所有具体参数值都满足一个特定的 DNF。对于字母表细化，即构建细化的守护条件，目前所采用的一种学习 DNF 的方法是按照文献[10]中的描述。

在文献[24]中，采用一般模式来在学习过程中递增扩展一个给定的字母表。然而在所提出的白箱情况（模型校验）下，无需细化守护条件和抽象。而是在当前考虑的字母表不够特别时，增加之前被忽略的作为输入的字母表符号。

11.7.1 参数化示例

文献[29]中考虑了具有多个整型参数的行为，并根据文献[25]中提出的 d 维空间平行轮廓加工盒子的组合学习技术来开发了一种字母表细化方法。这种实现方法的关键是将两种相互交织的分析进行明确分离，其中一种是利用 L^* 算法来获得自动机的结构，另一种是按照文献[25]来确定整型参数的阈值。

在这种设置下，抽象输入符号可由描述如图 11.9 所示的轴平行箱联合的线性方程来表示。接下来，仅在不涉及所用字母表细化方法具体细节的条件下简要介绍一些实验结果。在文献[29]中给出了所采用方法和结果的具体描述。

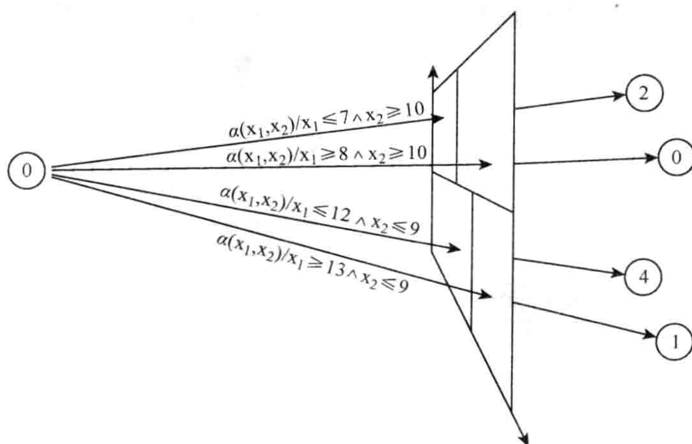


图 11.9 整型化参数操作

为估计参数化 Mealy 有限状态机性能变化如何影响学习算法中隶属度查询的行为，在此对于随机产生的不同性能的自动机进行了 4 组实验。在这 4 组学习实验的每个实验中，黑箱系统只有以下 1 个性能发生变化：行为个数、迁移次数、参数域大小和守护复杂性。所有生成的自动机都具有 6 个状态、1 个抽象行为和 1 个大小为 2 的输出字母表。对于每个结果分类，输出符号频率变化条件下产生了 60 个模型。

采用一种简单的方法来学习这些模型，首先将每个可能的参数值看作独立的输入符号（黑色），然后利用上述介绍的方法来细化仅所需的行为抽象集（白色）。假定所学习的系统在一定程度上是确定性的，并将非确定性发生看作一种非常粗糙抽象的唯一符号。图 11.10 表明了在对数尺度上不同实验中隶属度查询的平均次数。

- 图 11.10a 表明了参数个数增大时不同实验的结果。显然，原始算法中参数个数增加可导致隶属度查询次数的指数增大，而对于细化算法，则没有明显影响。
- 参数域（见图 11.10b）规模的增大可产生类似效果。
- 另一方面，增加迁移次数（即守护个数）对原始算法无任何影响，这毫不奇怪（见图 11.10c）。
- 在第 4 组实验中，守护的潜在复杂性逐步增大。图 11.10d 表明了潜在复杂性的增大对实际复杂性并没有显著影响。

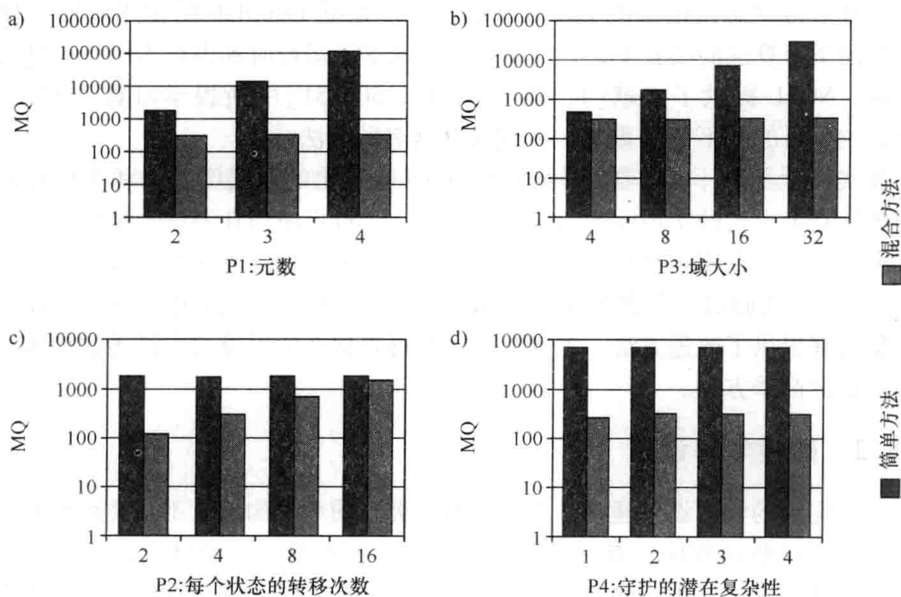


图 11.10 每组实验中隶属度查询（MQ）的平均次数

11.8 NGLL

在 LearnLib 工具中可获得最近 6 年来所积累的实践经验[49]。尽管该工具可配置且提供了一个丰富的算法和滤波器库，但构建一个合适的学习设置仍

并非易事。在某种程度上，这是因为可用的学习算法在实际中并不是工程通用的，且难以针对特殊案例进行编码，从而说明对于新的应用领域适用性有限。

NGLL[45]设计用于通过在扩展组件库的基础上支持基于模型的专用学习解决方案的构建来简化该任务。这些库包括处理实际系统所需的各种方法和工具，其中包括测试驱动程序、重置机制和抽象/细化技术。由于互联网使能，NGLL 还支持采用远程组件。因此，学习解决方案可由本地运行或世界上任何地方运行的组件混合组成，这样就具有了一个既对远程系统学习非常有用又可在分布式资源中灵活分配学习的特点。

11.8.1 基本技术

NGLL 是基于 jABC[53]的一种用于建模、开发、执行复杂应用程序和过程的面向服务的框架。NGLL 是最初设计用于对未知实际系统（通信系统、网络服务等）构建有限状态机模型的 LeanrLib 的扩展。根据 LearnLib 的使用经验可快速建立不同学习算法的实验平台，并对其学习成本、运行时和内存占用特点进行统计分析。NGLL 提供了文献[4，35，39，41，50，51]中所提学习算法的实现。然而，主要目的并不是（重复）实现这些著名的算法。

在实际学习中一个主要的障碍是由 Angluin 提出的隶属度查询和等价查询方面理想化查询形式的实现问题。这要求对于一个特定应用在测试和抽象方法上的相互影响，显然在实际中这很难实现。在这方面进行提高改善会导致完全重新设计 LearnLib，从而最终形成 NGLL。NGLL 提供了一种新的灵活度，并对性能分析和抽象处理提供了改进方法。关键在于一种可直接在学习设置中描述的构成学习过程的新的简单方法。

11.8.2 建模学习设置

一个完整的学习设置通常是由几种不同类型的组件构成：不同类型模型的学习算法、测试驱动程序、查询滤波器、缓存、模型输出、统计调查、抽象提供者、反例处理等。其中一些是可选的，而其他是必选的，许多这些组件本质上都是可重用的，且在不同应用背景下易于采用。NGLL 提供了这样一个组件集合并将其作为易用的构建块。由此使得学习过程设计人员可直观构成适用于应用的学习设置。

图 11.11 表明了在一个非常基本的学习场景下，典型的 NGLL 图形建模风格。如图中右侧一个典型学习设置所示，可非常容易地确定在大多数学习设置中经常发生的常用三阶段模式：

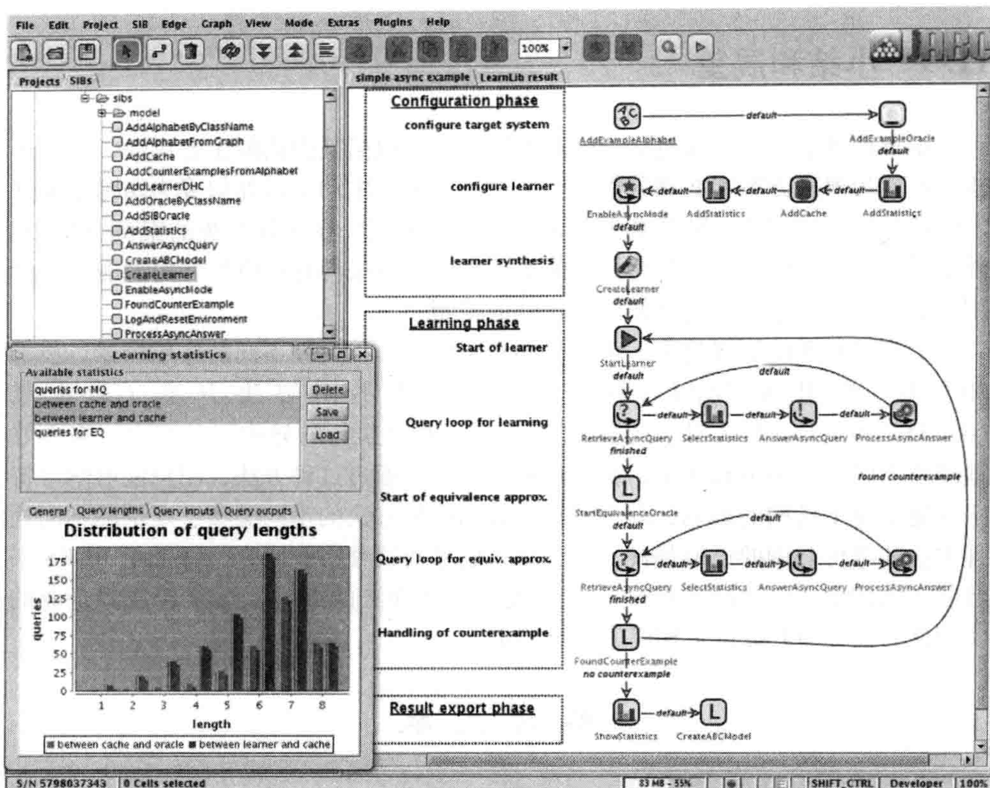


图 11.11　jABC 中一个简单的学习设置的执行服务逻辑图

- 学习过程首先是*配置阶段*，在此主要是在学习过程创建和开始之前，选择字母表和测试驱动程序。
- 接下来是学习过程的核心*学习阶段*，其特点是典型的 L^* 迭代，用于组织 SUL 中的基于测试的查询。这些迭代是在外推阶段构建的，并以构建完成假设自动机来结束，所谓的等价查询的（近似）实现，实际上就是搜索将假设自动机从 SUL 中分离的反例。如果搜索成功，将开始外推法的一个新阶段，用来处理反例所产生的所有结果。
- 否则，学习过程终止于第 3 阶段中某些后置处理，例如产生统计数据。

从根本上来讲，所有学习设置都按照该模式，通常由特定应用的细化来不断丰富。所设计的图形建模环境就是用于支持开发这种细化类型，例如组件重用、版本管理和评估。

学习设置由右侧主题集合的单个组件组成，如图 11.11 中的左上方所示。以这种方式设计的学习设置是可执行的。在窗口的左下方，BGLL 给出了执行学习设置时选择的统计数据，在这种情况下为查询长度的分配。

11.9 小结和展望

本章回顾了实际自动机学习中的本质问题，包括其面临的主要挑战、各种形式、可能的解决方案和案例分析说明，是为了表明主动学习如何成为实际系统开发中针对特定应用的一种功能强大的工具。事实上，在实际系统的案例分析中，外推了包含10000多个状态的行为模型，表明该方法具有规模扩展的潜能，以适用于更好地对应急行为进行面向运行时的控制。

初衷是对系统研究建立一个灵活的实验环境，以处理不同的应用场景、技术和工具，以开发NGLL。利用NGLL可将不同解决方案（如文献[4，35，39，41，50，51]）进行组合形成一种可执行的异构工具组合，来更好地理解不同设计决策和优化的影响与相互作用。通过这种方法，用户能够学习特定问题的模型结构的影响，其中包括参数和抽象方法的不同类型、反例的不同处理方法和等价查询的近似替代。为鼓励人们参与这种实验研究，降低了RERS平台的初始阈值[15]。RERS是反应式系统常规外推法的缩写，意味着强调企业的实用性，而NGLL提供了相应的关键技术。

参考文献

1. F. Aarts, J. Blom, T. Bohlin, Y.-F. Chen, F. Howar, B. Jonsson, M. Merten, R. Nagel, A. Sabetta, S. Soleimanifard, B. Steffen, J. Uijen, T. Wilk, and S. Windmuller. Establishing Basis for Learning Algorithms, 2010. Available at: http://hal.archives-ouvertes.fr/inria00464671/en/
2. F. Aarts, B. Jonsson, and J. Uijen. Generating models of infinite-state communication protocols using regular inference with abstraction. In *ICTSS*, pp. 188–204, 2010.
3. F. Aarts and F. Vaandrager. Learning I/O automata. In P. Gastin and F. Laroussinie, eds., *CONCUR 2010—Concurrency Theory, Volume 6269 of Lecture Notes in Computer Science*, pp. 71–85. Springer Berlin/Heidelberg, 2010.
4. D. Angluin. Learning regular sets from queries and counterexamples. *Information and Computation*, 75(2):87–106, 1987.
5. J. L. Balcázar, J. Díaz, and R. Gavaldà. Algorithms for learning finite automata from queries: A unified view. In *Advances in Algorithms, Languages, and Complexity*, pp. 53–72, 1997.
6. T. Berg, O. Grinchtein, B. Jonsson, M. Leucker, H. Raffelt, and B. Steffen. On the correspondence between conformance testing and regular inference. In M. Cerioli, ed., *Proceedings of the FASE '05, 8th International Conference on Fundamental Approaches to Software Engineering, Volume 3442 of Lecture Notes in Computer Science*, pp. 175–189. Springer Verlag, April 4–8, 2005.
7. T. Berg, B. Jonsson, M. Leucker, and M. Saksena. Insights to Angluin's learning. Technical Report 2003-039, Department of Information Technology, Uppsala University, August 2003.

8. T. Berg, B. Jonsson, and H. Raffelt. Regular inference for state machines with parameters. In L. Baresi and R. Heckel, eds., *Proceedings of the FASE '10, 13th International Conference on Fundamental Approaches to Software Engineering, Volume 3922 of Lecture Notes in Computer Science*, pp. 107–121. Springer Verlag, 2006.
9. T. Berg, B. Jonsson, and H. Raffelt. Regular inference for state machines using domains with equality tests. In J. L. Fiadeiro and P. Inverardi, eds., *Proceedings of the FASE '08, 11th International Conference on Fundamental Approaches to Software Engineering, Volume 4961 of Lecture Notes in Computer Science*, pp. 317–331. Springer Verlag, 2008.
10. A. Blum and S. Rudich. Fast learning of k-term DNF formulas with queries. In *Proceedings of the 24th ACM Symposium on Theory of Computing*, pp. 382–389. ACM, New York, 1992.
11. T. Bohlin and B. Jonsson. Regular inference for communication protocol entities. Technical Report, Department of Information Technology, Uppsala University, Sweden, 2009.
12. B. Bollig, J.-P. Katoen, C. Kern, and M. Leucker. Replaying play in and play out: Synthesis of design models from scenarios by learning. In O. Grumberg and M. Huth, eds., *Proceeding of 13th International Conference on Tools and Algorithms for the Construction and Analysis of Systems (TACAS '07), Held as Part of the Joint European Conferences on Theory and Practice of Software (ETAPS '07) Braga, Portugal, Volume 4424 of Lecture Notes in Computer Science*, pp. 435–450. Springer, 2007.
13. B. Bollig, J.-P. Katoen, C. Kern, and M. Leucker. Smyle: A tool for synthesizing distributed models from scenarios by learning. In F. van Breugel and M. Chechik, eds., *Proceedings of 19th International Conference on Concurrency Theory (CONCUR '08), Toronto, Canada, August 19–22, 2008, Volume 5201 of Lecture Notes in Computer Science*, pp. 162–166. Springer, 2008.
14. M. Broy, B. Jonsson, J.-P. Katoen, M. Leucker, and A. Pretschner. *Model-Based Testing of Reactive Systems, Volume 3472 of Lecture Notes in Computer Science*. Springer-Verlag New York, Secaucus, NJ, 2005.
15. Chair of Programming Systems, TU Dortmund, Department of Computer Science. RERS—A Challenge in Active Learning, 2010. http://leo.cs.tu-dortmund. de:8100/. Version from 20.06.2010.
16. T. S. Chow. Testing software design modeled by finite-state machines. *IEEE Transactions on Software Engineering*, 4(3):178–187, 1978.
17. E. M. Clarke, O. Grumberg, and D. A. Peled. *Model Checking*. The MIT Press, 1999.
18. J. M. Cobleigh, D. Giannakopoulou, and C. S. Pasareanu. Learning assumptions for compositional verification. In *Proceedings of the TACAS '03, 9th International Conference on Tools and Algorithms for the Construction and Analysis of Systems, Volume 2619 of Lecture Notes in Computer Science*, pp. 331–346. Springer Verlag, 2003.
19. D. Combe, C. de la Higuera, and J.-C. Janodet. Zulu: An interactive learning competition. In *Proceedings of FSMNLP 2009*, 2010 (to appear).
20. F. Denis, A. Lemay, and A. Terlutte. Residual finite state automata. *Fundamenta Informaticae*, 51(4):339–368, 2002.

21. J. Esparza, M. Leucker, and M. Schlund. Learning workflow petri nets. In *Proceedings of the 31st International Conference on Application and Theory of Petri Nets and Other Models of Concurrency (Petri Nets'10), Lecture Notes in Computer Science*. Springer, 2010.
22. S. Fujiwara, G. von Bochmann, F. Khendek, M. Amalou, and A. Ghedamsi. Test Selection based on finite state models. *IEEE Transactions on Software Engineering*, 17(6):591–603, 1991.
23. H. Garavel. Open/caesar: An open software architecture for verification, simulation, and testing. In *Proceedings of the 4th International Conference on Tools and Algorithms for Construction and Analysis of Systems*, pp. 68–84. Springer-Verlag, London, UK, 1998.
24. M. Gheorghiu, D. Giannakopoulou, and C. S. Pasareanu. Refining interface alphabets for compositional verification. In *TACAS*, pp. 292–307, 2007.
25. P. W. Goldberg, S. A. Goldman, and H. D. Mathias. Learning unions of boxes with membership and equivalence queries. In *Proceedings of COLT '94, 7th Annual Conference on Computational Learning Theory*, pp. 198–207, ACM, New York, 1994.
26. O. Grinchtein, B. Jonsson, and P. Pettersson. Inference of event-recording automata using timed decision trees. In *Proceedings of CONCUR 2006, 17th International Conference on Concurrency Theory*, pp. 435–449, 2006.
27. A. Hagerer, H. Hungar, O. Niese, and B. Steffen. Model generation by moderated regular extrapolation. *Lecture Notes in Computer Science*, pp. 80–95, 2002.
28. A. Hagerer, T. Margaria, O. Niese, B. Steffen, G. Brune, and H.-D. Ide. Efficient regression testing of CTI-systems: Testing a complex call-center solution. Annual review of communication. *International Engineering Consortium (IEC)*, 55:1033–1040, 2001.
29. F. Howar. Inferenz parametrisierter moore-automaten. Master's thesis, Technische Universität Dortmund, Fakultät für Informatik, Lehrstuhl für Programmiersysteme, 2009.
30. F. Howar, B. Steffen, and M. Merten. From ZULU to RERS—Lessons learned in the ZULU challenge. In T. Margaria and B. Steffen, eds., *ISoLA (1), Volume 6415 of Lecture Notes in Computer Science*, pp. 687–704. Springer, 2010.
31. F. Howar, B. Steffen, and M. Merten. Automata learning with automated alphabet abstraction refinement. In R. Jhala and D. A. Schmidt, eds., *VMCAI, Volume 6538 of Lecture Notes in Computer Science*, pp. 263–277. Springer, 2011.
32. F. Howar, B. Steffen, and M. Merten. Automata learning with automated alphabet abstraction refinement. In *Twelfth International Conference on Verification, Model Checking, and Abstract Interpretation*, 2011.
33. H. Hungar, T. Margaria, and B. Steffen. Model generation for legacy systems. In M. Wirsing, A. Knapp, and S. Balsamo, eds., *RISSEF, Volume 2941 of Lecture Notes in Computer Science*, pp. 167–183. Springer, 2002.
34. H. Hungar, T. Margaria, and B. Steffen. Test-based model generation for legacy systems. In *Test Conference, 2003. Proceedings. ITC 2003. International, Volume 1*, pp. 971–980. September 30–October 2, 2003.
35. H. Hungar, O. Niese, and B. Steffen. Domain-specific optimization in automata learning. In W. A. Hunt, Jr. and F. Somenzi, eds., *Proceedings of the 15th Interna-*

tional Conference on Computer Aided Verification, Volume 2725 of Lecture Notes in Computer Science, pp. 315–327. Springer Verlag, July 2003.

36. H. Hungar and B. Steffen. Behavior-based model construction. *International Journal on Software Tools for Technology Transfer*, 6(1):4–14, 2004.
37. V. Issarny, B. Steffen, B. Jonsson, G. S. Blair, P. Grace, M. Z. Kwiatkowska, R. Calinescu, P. Inverardi, M. Tivoli, A. Bertolino, and A. Sabetta. CONNECT challenges: Towards emergent connectors for eternal networked systems. In *ICECCS*, pp. 154–161, 2009.
38. G. Jung, T. Margaria, C. Wagner, and M. Bakera. Formalizing a methodology for design- and runtime self-healing. In *Engineering of Autonomic and Autonomous Systems, IEEE International Workshop*, 0:106–115, 2010.
39. M. J. Kearns and U. V. Vazirani. *An Introduction to Computational Learning Theory*. MIT Press, Cambridge, MA, 1994.
40. M. Z. Kwiatkowska, G. Norman, D. Parker, and H. Qu. Assume-guarantee verification for probabilistic systems. In *TACAS*, pp. 23–37, 2010.
41. O. Maler and A. Pnueli. On the learnability of infinitary regular sets. *Information and Computation*, 118(2):316–326, 1995.
42. T. Margaria, O. Niese, H. Raffelt, and B. Steffen. Efficient test-based model generation for legacy reactive systems. In *HLDVT '04: Proceedings of the High-Level Design Validation and Test Workshop, 2004. Ninth IEEE International*, pp. 95–100. IEEE Computer Society, Washington, DC, 2004.
43. T. Margaria, H. Raffelt, and B. Steffen. Analyzing second-order effects between optimizations for system-level test-based model generation. In *Test Conference, 2005. Proceedings. ITC 2005. IEEE International*. IEEE Computer Society, November 2005.
44. T. Margaria, H. Raffelt, and B. Steffen. Knowledge-based relevance filtering for efficient system-level test-based model generation. *Innovations in Systems and Software Engineering*, 1(2):147–156, 2005.
45. M. Merten, B. Steffen, F. Howar, and T. Margaria. Next Generation LearnLib. In *Seventeenth International Conference on Tools and Al-Gorithms for the Construction and Analysis of Systems*, 2011.
46. C. S. Pasareanu, D. Giannakopoulou, M. G. Bobaru, J. M. Cobleigh, and H. Barringer. Learning to divide and conquer: Applying the L* algorithm to automate assume-guarantee reasoning. *Formal Methods in System Design*, 32(3):175–205, 2008.
47. H. Raffelt, T. Margaria, B. Steffen, and M. Merten. Hybrid test of web applications with webtest. In *TAV-WEB '08: Proceedings of the 2008 Workshop on Testing, Analysis, and Verification of Web Services and Applications*, pp. 1–7. ACM, New York, 2008.
48. H. Raffelt, M. Merten, B. Steffen, and T. Margaria. Dynamic testing via automata learning. *International Journal on Software Tools for Technology Transfer*, 11(4):307–324, 2009.
49. H. Raffelt, B. Steffen, T. Berg, and T. Margaria. LearnLib: A framework for extrapolating behavioral models. *International Journal on Software Tools for Technology Transfer*, 11(5):393–407, 2009.
50. R. L. Rivest and R. E. Schapire. Inference of finite automata using homing sequences. *Information and Computation*, 103(2):299–347, 1993.

51. M. Shahbaz and R. Groz. Inferring mealy machines. In *FM '09: Proceedings of the 2nd World Congress on Formal Methods*, pp. 207–222. Springer Verlag, Berlin, Heidelberg, 2009.
52. M. Shahbaz, K. Li, and R. Groz. Learning parameterized state machine model for integration testing. In *Proceedings of the 31st Annual International Computer Software and Applications Conference, Volume 2*, pp. 755–760. IEEE Computer Society, Washington, DC, 2007.
53. B. Steffen, T. Margaria, R. Nagel, S. Jörges, and C. Kubczak. *Model-Driven Development with the jABC, Volume 4383 of LNCS*, pp. 92–108. Springer Berlin/Heidelberg, 2006.

国际视野　科技前沿

国际信息工程先进技术译丛

《工业关键系统的形式化方法：应用综述》
《基于片上去耦电容的配电网络》（原书第2版）
《智能摄像机》
《车载系统和安全的数字信号处理》
《基于视觉的自主机器人导航》
《自主式传感器系统的能量收集——设计、分析以及实践应用》
《移动云计算：无线、移动及社交网络中分布式资源的开发利用》
《Android系统安全与攻防》
《内容分发网络》
《认知视角下的无线传感器网络》
《计算机网络仿真OPNET实用指南》
《移动无线信道》（原书第2版）
《LTE-Advanced：面向IMT-Advanced的3GPP解决方案》
《声学成像技术及工程应用》
《认知无线电通信与组网：原理与应用》
《LTE/SAE网络部署实用指南》
《网络性能分析原理与应用》
《云连接与嵌入式传感系统》
《IP地址管理原理与实践》
《自组织网络：GSM，UMTS和LTE的自规划、自优化和自愈合》
《实现吉比特传输的60GHz无线通信技术》
《LTE自组织网络（SON）：高效的网络管理自动化》
《UMTS中的LTE：向LTE-Advanced演进》（原书第2版）
《无线传感器及执行器网络》
《UMTS中的WCDMA－HSPA演进及LTE》（原书第5版）
《认知无线电网络》
《网络融合 ——服务、应用、传输和运营支撑》
《UMTS中的LTE:基于OFDMA和SC-FDMA的无线接入》
《高性能微处理器电路设计》
《吉规模集成电路互连工艺及设计》
《高级电子封装》（原书第2版）
《基于4G系统的移动服务技术》
《移动无线传感器网 ——技术、应用和发展方向》
《UMTS蜂窝系统的Q0S与Q0E管理》
《UMTS-HSDPA系统的TCP性能》
《基于射频工程的UMTS空中接口设计与网络运行》
《未来UMTS的体系结构与业务平台：全IP的3GCDMA网络》
《环境网络：支持下一代无线业务的多域协同网络》
《基于蜂窝系统的IMS—融合电信领域的V0IP演进》

WILEY



机械工业出版社微信服务号



上架指导　工业技术／自动化／计算机技术



ISBN 978-7-111-48521-6　定价：69.00元